Advances in Fuzzy Systems

High Performance Fuzzy Systems for Real World Problems

Guest Editors: Oscar Montiel Ross, Roberto Sepúlveda Cruz, Oscar Castillo, and Alper Basturk

High Performance Fuzzy Systems for Real World Problems

High Performance Fuzzy Systems for Real World Problems

Guest Editors: Oscar Montiel Ross, Roberto Sepúlveda Cruz, Oscar Castillo, and Alper Basturk

Editorial Board

Contents

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 316187, 2 pages
doi:10.1155/2012/316187

Editorial

High Performance Fuzzy Systems for Real World Problems

Oscar Montiel Ross,[1] Roberto Sepúlveda Cruz,[1] Oscar Castillo,[2] and Alper Basturk[3]

[1] *CITEDI, Instituto Politécnico Nacional, Tijuana, BC, Mexico*
[2] *Tijuana Institute of Technology, P.O. Box 4207, Chula Vista, CA 91909, USA*
[3] *Department of Computer Engineering, Erciyes University, 38039 Kayseri, Turkey*

Correspondence should be addressed to Oscar Montiel Ross, o.montiel@ieee.org

Received 11 December 2012; Accepted 11 December 2012

Nowadays, the importance of developing high-performance computers (HPC) and efficient algorithms that take advantage of new technologies is rapidly growing because the diversity and complexity of mathematical models that need simulations are increasing; as well as there are more technical problems that require to process data at high speed. HPC is an invaluable tool for analysts, engineers, and scientist because it offers the resources that they need to make vital decisions, to speed up research and developments, to promote product innovations, and to reduce time to market. With respect to fuzzy logic, there are a wide variety of fuzzy-based applications for consumer electronics that must satisfy real time constrains for processing data at high speed, as well as large quantities of data containing uncertainty from different sources that often cannot be adequately modeled and/or handled by type-1 fuzzy sets.

This special issue contains five papers that deal with important topics that are contributions to confront the above challenges in an original and efficient way. One paper is about a new voltage-input, current-output programmable membership function generator circuit using CMOS technology. Two papers are about improving performance of fuzzy systems using Field Programmable Gate Array (FPGA); in the proposals, the Fusion and SmartFusion FPGAs from Actel are used; moreover, in these papers the use of embedded hard and soft processors, intellectual property cores (IP-Cores), programming using C language and VHDL is illustrated. Two papers address the problem of improving performance of type 2 fuzzy systems.

In the paper entitled "*A novel programmable CMOS fuzzifiers using voltage-to-current converter circuit*" by K. Abdulla and M. Azeem, a new voltage-input, current-output programmable membership function generator circuit (MFC) using CMOS technology is presented. It employs a voltage-to-current converter to provide the required current bias for the membership function circuit. The proposed MFC has several advantageous features. This MFC can be reconfigured to perform triangular, trapezoidal, S-shape, Z-shape, and Gaussian membership forms. This membership function can be programmed in terms of its width, slope, and its center locations in its universe of discourse. The easily adjustable characteristics of the proposed circuit and its accuracy makes it suitable for embedded systems and industrial control applications. The proposed MFC is designed using the spice software, and obtained simulation results are reported in the paper.

The paper entitled "*Designing high-performance fuzzy controllers combining IP cores and soft processors*" by O. Montiel et al. presents a methodology to integrate a fuzzy coprocessor described in VHDL (VHSIC Hardware Description Language) to a soft processor embedded into an FPGA. The aim is to increase the throughput of the whole system; since the controller uses parallelism at the circuitry level for high-speed-demanding applications, the rest of the application can be written in C/C++. The ARM 32-bit soft processor, which allows sequential and parallel programming, is used. The FLC coprocessor incorporates a tuning method that allows manipulating the system response with just modifying one parameter. The authors show experimental results using a fuzzy PD+I controller as the embedded coprocessor. Comparative results for different hardware platforms such as a desktop personal computer, the

Spartan 3 FPGA, the Virtex 5 FPGA, and the Atmel AVR 8-bits microcontroller are presented.

In the paper entitled "*WLAN cell handoff latency abatement using an FPGA fuzzy logic algorithm implementation*," R. Sepúlveda et al. present a predictive fuzzy logic controller FPGA implementation to speed up the processing time to reduce the channel scanning process of a WiFi communication system to a tenth of the standard time. The IEEE 802.11n standard yields data rates up to 450 Mbps, and the 802.11e standard ensures proficient QoS for real-time applications. Still in need of better performance, multicell environments that provide extended coverage allow the mobile station nomadic passage beyond a single cell by means of cell dissociation-association process known as handoff. This process poses a challenge for real-time applications like voice over IP (150 ms maximum delay) and video (200–400 ms) sessions to give the user a seamless cell-crossing without data loss or session breakage. The algorithm of the fuzzy controller is implemented in C language. Experimental results using the FPGA SmartFusion are provided.

In the paper "*Speedup of interval type 2 fuzzy logic systems based on GPU for robot navigation*," L. Ngo et al. illustrate how to install interval type-2 FLS (IT2-FLS) on a graphics processing unit (GPU) and the use of nVIDIA's Compute Unified Device Architecture (CUDA); the authors show experiments for obstacle avoidance behavior of robot navigation. They carry out FLSs analysis in order to take advantage of GPUs processing capabilities to speed up IT2-FLSs. They demonstrate that use of the computer CPU outperforms the GPU for small systems and conclude that as the number of rules and sample rate grow, the GPU outperforms the CPU. They show that there is a switch point in the performance ratio which indicates when the GPU is more efficient than the CPU. GPU runs approximately 30 times faster on the computer.

In the paper entitled "*A hybrid model through the fusion of type-2 fuzzy logic systems and sensitivity-based linear learning method for modeling PVT properties of crude oil systems*," A. Selamat et al. present a proposal for modeling pressure-volume-temperature (PVT) properties, which are crucial for geophysics and petroleum engineers, namely, for utilization in material balance calculations, inflow performance calculations, well log analysis, determining oil reserve estimations, and the amount of oil that can be recovered, the flow rate of oil or gas, and the simulations on reservoir outputs. In this proposal, a hybrid system based on a sensitivity-based linear learning method (SBLLM) that has been recently used as a predictive tool because its unique characteristics and performance, particularly its high stability and consistency during predictions, in combination with a type-2 fuzzy logic system (FLS) to handle uncertainties in reservoir data is used. This work presents comparative studies to compare the performance of the newly proposed T2-SBLLM hybrid system with each of the constituent type-2 FLS and SBLLM. The empirical results from simulation show that the proposed T2-SBLLM hybrid system has greatly improved upon the performance of SBLMM, while also maintaining a better performance about of the type-2 FLS.

Acknowledgments

We would like to thank the authors for their excellent works that contain important contributions in the field of HPC, as well as the reviewers for their important role in the review process of the papers.

Oscar Montiel Ross
Roberto Sepúlveda Cruz
Oscar Castillo
Alper Basturk

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 359429, 19 pages
doi:10.1155/2012/359429

Research Article

A Hybrid Model through the Fusion of Type-2 Fuzzy Logic Systems and Sensitivity-Based Linear Learning Method for Modeling PVT Properties of Crude Oil Systems

Ali Selamat,[1] Sunday Olusanya Olatunji,[1] and Abdul Azeez Abdul Raheem[2]

[1] *Intelligent Software Engineering Laboratory, Faculty of Computer Science and Information Systems, University of Technology Malaysia, 81310 Skudai, Johor Bahru, Malaysia*
[2] *Centre for Petroleum and Minerals, The Research Institute, King Fahd University of Petroleum and Minerals (KFUPM), P.O. Box 1105, Dhahran 31261, Saudi Arabia*

Correspondence should be addressed to Ali Selamat, aselamat@utm.my

Received 29 March 2012; Revised 8 September 2012; Accepted 10 September 2012

Academic Editor: Oscar Montiel Ross

Sensitivity-based linear learning method (SBLLM) has recently been used as a predictive tool due to its unique characteristics and performance, particularly its high stability and consistency during predictions. However, the generalisation capability of SBLLM is sometimes limited depending on the nature of the dataset, particularly on whether uncertainty is present in the dataset or not. Since it made use of sensitivity analysis in relation to the data sets used, it is surely very prone to being affected by the nature of the dataset. In order to reduce the effects of uncertainties in SBLLM prediction and improve its generalisation ability, this paper proposes a hybrid system through the unique combination of type-2 fuzzy logic systems (type-2 FLSs) and SBLLM; thereafter the hybrid system was used to model PVT properties of crude oil systems. Type-2 FLS has been choosen in order to better handle uncertainties existing in datasets beyond the capability of type-1 fuzzy logic systems. In the proposed hybrid, the type-2 FLS is used to handle uncertainties in reservoir data so that the cleaned data from type-2 FLS is then passed to the SBLLM for training and then final prediction using testing dataset follows. Comparative studies have been carried out to compare the performance of the newly proposed T2-SBLLM hybrid system with each of the constituent type-2 FLS and SBLLM. Empirical results from simulation show that the proposed T2-SBLLM hybrid system has greatly improved upon the performance of SBLLM, while also maintaining a better performance above that of the type-2 FLS.

1. Introduction

Hybrid computational intelligence is any effective combination of intelligent techniques that performs superior or in a competitive way to simple standard intelligent techniques. The increased popularity of hybrid intelligent systems in recent times lies in the extensive success of these systems in many real-world complex problems [1]. Also, it is an established fact that every approach has its strengths and weaknesses; hence the need for hybrid models that is able to combine the strengths of the individual techniques while complementing the weaknesses of one method with the strength of the other. Therefore, this work seeks to take advantage of the unique capability of type-2 FLS, in modelling uncertainties, to improve the performance of the sensitivity based linear learning method (SBLLM) in order to further boost the generalisation ability of SBLLM even in the face of uncertainties.

Type-2 fuzzy logic has been generally acknowledged as being better and ideal for uncertainty modeling [2–8]. Recently, type-2 FLS have been proposed as a novel framework for both classification and prediction in order to handle all forms of uncertainties [8, 9]. It is able to handle uncertainties that include those in measurements and data used to calibrate the parameters. It has been used in several fields and the results have been promising and very encouraging [10–12]. Therefore, there is a possibility that the type-2 FLS can handle uncertainty in reservoir data [13] as

the type-2 fuzzy logic has been specifically invented to deal with all forms of uncertainties [8] that are inherent in our day to day natural encounters and mode of reasoning.

Sensitivity based linear learning method (SBLLM) has been recently introduced as a learning technique for two-layer feedforward neural networks based on sensitivity analysis that uses a linear training algorithm for each of the two layers [14]. It was introduced in order to alleviate some of the limitations of the classical ANN. This algorithm tends to provide good generalization performance at extremely fast learning speed, while in addition, it gives the sensitivities of the sum of squared errors with respect to the input and output data without extra computational cost. It is very stable in performance as its learning curve stabilizes soon and behaves homogeneously not only if we consider just the end of the learning process, but also during the whole process, in such a way that very similar learning curves were obtained for all iterations of different experiments [14, 15]. Unfortunately, just like the classical ANN, SBLLM is unable to adequately model uncertainties in real life data. As a result of SBLLM inability to handle uncertainty, it will be a good contribution to seek to improve its performance through the use of type-2 FLS as its precursor, in a hybrid arrangement, for uncertainty handling, in order to take advantage of its unique capabilities. Therefore in this paper, we propose a hybrid approach that will combine the unique attributes of type-FLS with those of sensitivity based linear learning method (SBLLM) by way of improving SBLLM performance in order to achieve better generalisation ability in all situations including uncertainty oriented environment.

Characterization of reservoir fluids plays a very crucial role in developing a strategy on how to produce and operate a reservoir. Pressure-volume-temperature (PVT) properties are very crucial for geophysics and petroleum engineers, namely for the utilization in material balance calculations, inflow performance calculations, well log analysis, determining oil reserve estimations, and the amount of oil that can be recovered, the flow rate of oil or gas, and the simulationson reservoir outputs. The phase and volumetric behaviour of petroleum reservoir fluids is referred to as PVT [16, 17].

PVT properties include formation volume factor (FVF), solution gas-oil ratio (GOR), solution oil-gas ratio (OGR), liquid specific gravity, American petroleum institute (API) specific gravity, gas specific gravity, bubble-point pressure, saturation pressure, and so forth stated by [18]. Among those PVT properties, the bubble-point pressure (P_b) and the oil formation factor (B_{ob}) are the most important, because they are the most essential factors in reservoir and production computations [18]. The more the preciseness of estimating these properties, the better the calculations involved in reservoir simulation, production, and field development. Bubble-point pressure (P_b) is the pressure at which gas first begins to come out of the solution at constant temperature, while oil formation volume factor (B_{ob}) is defined as the volume of reservoir oil that would be occupied by one stock tank barrel oil plus any dissolved gas at the bubble point pressure and reservoir temperature as stated in [17, 19–21].

Ideally, these properties are determined from laboratory studies on samples collected from the bottom of the wellbore or on the surface. However, such experimental data is very costly to obtain and the accuracy of the result is critical and not often known in advance. One of the solutions is to use the empirically derived correlations, which has been developed using equation of state (EOS), linear/non-linear statistical regression, or graphical techniques [17, 21]. Unfortunately, these correlations are constrained by several limitations, namely, the equation of state requires the extensive knowledge of the detailed compositions of the reservoir fluids and the determination of such quantities is expensive and time consuming; the accuracy of the EOS depends heavily on the nature of the fluid, on the type of equation selected, and on the operator-dependent tuning procedures. This method also involves several numerical computations. To overcome the shortcomings associated with the earlier correlation methods, researchers made use of artificial intelligence based methods foremost of which is the classical artificial neural network (ANN) and its variants. But still, the developed neural networks correlations often do not perform to expectations and are bedeviled with shortcomings that include, among others, instability, with its non homogeneous nature such that very different learning curves were obtained for different repeat of same experiment, and its characteristic low speed operation.

Researchers have done their best to address and overcome these problems of ANN. As a result, several variants of ANN and other methods like support vector machines (SVM) and functional networks (FN) have been proposed and used [21, 22], yet each has its limitations that still call for further research of this nature, particularly their inability to handle uncertainties and the need to ensure stability and consistency in predictions.

It is an established fact that geosciences disciplines are not clear-cut and, most of the time, are associated with uncertainties [13], hence the need for fuzzy logic based systems, particularly the newly introduced type-2 fuzzy logic system (type-2 FLS) that is able to adequately account for all forms of uncertainties [8]. For instance, prediction of core parameters from well log responses is difficult and is usually associated with uncertainties. Earlier methods try to minimize and ignore these uncertainties [13], while type-2 fuzzy logic derives useful information from the uncertainties and uses it as a good selection of parameters for increasing the accuracy of the predictions. SBLLM on its own is able to deploy its sensitivity analysis to ensure stable and consistent results always. Thus a combination of these unique methods in a hybrid arrangement will go a long way in improving the prediction accuracy while ensuring stability and consistency, which are requisite of good prediction system. This will definitely be in line with the often reported successes of hybrid systems as a result of the unique combination of methods that takes advantage of each constituent member while avoiding the shortcomings of each.

Therefore, this paper investigate the feasibility of using type-2 FLS as a pre-cursor to improve the generalisation ability of SBLLM in the face of uncertainty during prediction, in a hybrid framework setting; we develop a new hybrid model based on type-2 FLS and SBLLM and then use it for predicting PVT properties that include specifically bubble

point pressure (P_b) and oil formation volume factor (B_{ob}) using different standard databases of four input parameters, namely, solution gas-oil ratio, reservoir temperature, oil gravity, and gas relative density. We then investigate how individual constituent methods compare in their forecasting performance with the proposed hybrid model. Empirical results from simulations demonstrated that the proposed hybrid scheme produced better generalization performance, with high stability and consistency, which are requisite of good prediction models, compared to each of the constituent parts, particularly SBLLM.

The rest of this paper is organized as follows. Section 2 presents a review of related researches and Section 3 presents the proposed hybrid model and its constituent parts. Section 4 contains the empirical study and implementation process. Results and discussions are presented in section5. The conclusion and future work recommendations are provided in Section 6.

2. Related Research

In the past few decades, engineers realized the importance of developing and using empirical correlations for PVT properties. The development of correlations for PVT calculations has been the subject of extensive research, resulting in a large volume of publications. In this section, we briefly review the most common empirical PVT correlations and the related prediction approach together with the measurement techniques that have been used in forecasting these PVT properties.

2.1. Common Empirical Models and Evaluation Studies. Standing [23] presented correlations for bubble point pressure and for oil formation volume factor. The correlations were based on laboratory experiments carried out on 105 samples from 22 different crude oils in California. Glaso [24] developed the Glasoempirical correlation for formation volume factor using 45 oil samples from North Sea hydrocarbon mixtures. Al-Marhoun [25] published his second correlation for oil formation volume factor. The correlation was developed with 11,728 experimentally obtained formation volume factors at, above, and below bubble point pressure. The data set represented samples from more than 700 reservoirs from all over the world, mostly from Middle East and North America. For more empirical correlation-related work, discussion, applications, and comparative studies; interested readers can see [26–42].

2.2. Predicting PVT Properties Based on Artificial Neural Networks. Artificial Intelligence schemes have been increasingly used in the field of PVT properties and other fields in oil and gas industry during the last few decades, the most popular of which are the neural networks. Artificial neural networks are parallel-distributed information processing models that can recognize highly complex patterns within available data. In recent years, neural network have gained popularity in petroleum applications. Many authors have discussed the applications of neural networks in petroleum engineering; see [16, 19, 43–47] for details. Recently, it has been shown in both machine learning and data mining communities that artificial neural networks have the capacity to learn complex linear/nonlinear relationships amongst input and output data. The most common widely used neural network in literature is known as the feed forward neural networks with back propagation training algorithm [48]. This type of neural networks is an excellent computational intelligence modelling scheme in both prediction and classification tasks. Recently, feed forward neural networks were used to predict the PVT correlations, [16, 49–51].

The authors in [52] introduced a novel approach for predicting the complete PVT behavior of reservoir oils and gas condensates using a noniterative approach. The method uses key measurements that can be performed rapidly either in the lab or at the well site as input to a neural network, while in [51], two neural networks were trained separately to estimate the bubble point pressure (P_b) and oil formation volume factor (B_{ob}), respectively. The input data were solution gas-oil ratio, reservoir temperature, oil gravity, and gas relative density, while making use of two hidden layers (2HL) neural networks: the first neural network, (4-8-4-2) to predict the bubble point pressure and the second neural network, (4-6-6-2) to predict the oil formation volume factor. Both neural networks were built using a data set of size 520 observations from Middle East area. The input data set was divided into a training set of 498 observations and a testing set of 22 observations.

The authors in [19] used the feedforward learning scheme with log sigmoid transfer function in order to estimate the formation volume factor at the bubble point pressure, using data published in [50], while the authors in [53] developed two new models to predict the bubble point pressure, and the oil formation volume factor at the bubble-point pressure for Saudi crude oils. The models were based on artificial neural networks, and developed using 283 unpublished data sets collected from different Saudi fields. Recently, [46] made use of neural network to predict the PVT properties. As usual, they were confronted with the generic problems of the standard neural network like local minima convergence problem, trial and error syndrome, instability, and inconsistency.

For further works on the utilization of neural networks and its variants, such as radial basis function network, support vector machines, and abductive network, for predicting PVT properties, interested readers can refer to [16, 17, 21, 47, 49, 54–57].

We have noted that most of the reported cases of the use of fuzzy logic in modeling reservoir properties are restricted to the classical fuzzy logic (also known as type-1 fuzzy logic). However, type-1 fuzzy logic systems have recently been found inadequate for handling all forms of uncertainties [5, 8, 9, 58]. In response to this, type-2 fuzzy logic systems have been introduced as better computational intelligence approach for both prediction and classification to handle all forms of uncertainties [5]. The unique feature and advantage of type-2 fuzzy logic systems and those of sensitivity based linear learning methods (SBLLM) motivated the idea of this work.

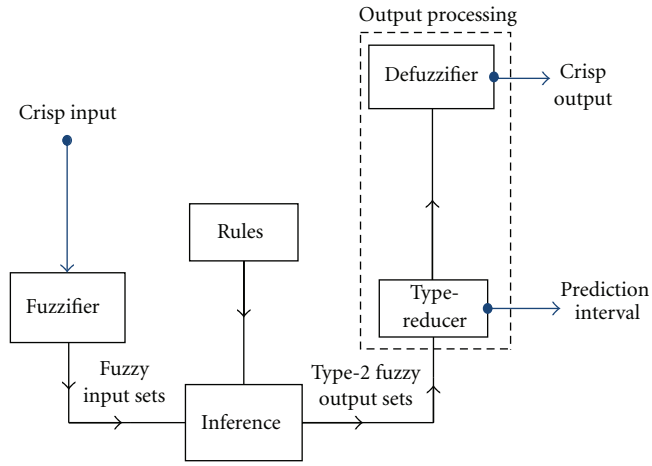


Figure 1: Schematic diagram of type-2 fuzzy logic systems (type-2 FLS).

In order to further boost the accuracy of predictions, particularly in this germane field of reservoir characterization where the need for accurate prediction is highly desirable, we have proposed a better and more reliable hybrid scheme that will be able to adequately model uncertainty in reservoir data and make accurate predictions while ensuring stable and consistent results.

In this regard, this work seeks to develop a new hybrid scheme based on type-2 fuzzy logic system and sensitivity based linear learning methods (SBLLM). These combinations have been choosing due to the fact that: type-2 fuzzy logic system is able to model all forms of uncertainties and sensitivity based linear learning methods (SBLLM) have a unique generalization ability coupled with higher stability and consistency. With these, the proposed hybrid model will be able to make use of these unique combinations for effective uncertainties handling while ensuring robust, consistent, and accurate performance.

3. The Proposed Hybrid Model and Its Constituent Frameworks

The proposed hybrid system is composed of type-2 fuzzy logic system and sensitivity based linear learning methods (SBLLM) learning schemes, uniquely combined together to form a better performing hybrid scheme.

3.1. Type-2 Fuzzy Logic System (Type-2 FLS). Type-2 adaptive fuzzy inference systems is an adaptive network that learns the membership functions and fuzzy rules, from data, in a fuzzy system based on type 2 fuzzy sets; see [8, 59] for details. "Type-2 fuzzy sets are fuzzy sets whose grades of membership are themselves fuzzy. They are intuitively appealing because grades of membership can never be obtained precisely in practical situations" [60]. Type-2 fuzzy sets can be used in situations where there is uncertainty about the membership grades themselves, for example, an uncertainty in the shape of the membership function or in some of its parameters. Consider the transition from ordinary sets to fuzzy sets; when we cannot determine the membership of an element in a set as 0 or 1 we use fuzzy sets of type-1. Similarly, when the situation is very fuzzy that we have difficulty in determining the membership grade as a crisp number in $[0, 1]$, we use fuzzy sets of type-2. Thus, in general, "a fuzzy set is of type n, $n = 2, 3, \ldots$ if its membership function ranges over fuzzy sets of type $n - 1$" [61].

Generally, a type-2 fuzzy logic system contains five components: fuzzifier, rules, inference engine, type-reducer, and defuzzifier that are interconnected as in Figure 1.

The fuzzifier takes the input parameters values as inputs. The output of the fuzzifier is the fuzzified measurements which will be the input to the inference engine. The resultant of the inference engine is type-2 fuzzy output sets which can be reduced to type-1 fuzzy set by the type reducer. The type reduced fuzzy set in this model is an interval set which gives the predicted external attribute measurement as a possible range of values. The defuzzifier calculates the average of this interval set to produce the predicted crisp external attribute measurement.

3.1.1. Inferencing in the Type-2 FLS. Fuzzy inference engine combines the fired fuzzy rules and map inputs into type-2 output fuzzy sets. Generally a type-2 FLS is a fuzzy logic system in which at least one of the fuzzy sets used in the antecedent and/or consequent parts and each rule inference output is a type-2 fuzzy set. Consider a type-2 Mamdani FLS [8] having n inputs $x_1 \in X_1, \ldots, x_n \in X_n$ and one output $y \in Y$. The rule base contains L type-2 fuzzy rules expressed in the following form:

$$R^{\mathrm{l}}: \; IF \; x_1 \text{ is } F_1^1 \text{ and } x_2 \text{ is } F_2^1 \text{ and} \ldots \text{and } x_{\mathrm{n}} \text{ is } F_p^1 \text{ THEN } y \text{ is } G^{\mathrm{l}}, \tag{1}$$

where $l = 1, 2, \ldots, L$, F_i^l and G^l are type-2 fuzzy set.

This rule represents a type-2 fuzzy relation between the input space $X \in X_1 \times X_2 \times \cdots \times X_n$, and the output space

(1) Arrange y_z in ascending order with $y_1 \le y_2 \le \cdots \le y_z$
(2) Set $\theta_z = h_z$ for $z = 1, 2, \ldots, Z$; and compute $y' = \sum_{z=1}^{Z} y_z\theta_z / \sum_{z=1}^{Z} \theta_z$
(3) Find $e \in [1, Z-1]$ such that $y_e \le y' \le y_{e+1}$
(4) Set $\theta_z = h_z - \Delta_z$ for $z \le e$ and $\theta_z = h_z + \Delta_z$ for $z > e$, and compute: $y'' = \sum_{z=1}^{Z} y_z\theta_z / \sum_{z=1}^{Z} \theta_z$
(5) Stop if $y'' = y'$, otherwise set $y' = y''$ and return to step 3

Algorithm 1: Karnik-Mendel algorithm to compute y_r^t.

Y of the system. We denote the membership function of this Type-2 relation as:

$$\mu_{F_1^l \times \cdots \times F_n^l} \longrightarrow G^l(x, y), \tag{2}$$

where $F_1^l \times \cdots \times F_n^l$ denotes the Cartesian product of F_1^l, $F_2^l, \ldots, F_n^l$ and $x = \{x_1, x_2, \ldots, x_n\}$.

The antecedents in the fuzzy rules are connected by using the meet operation, the firing strength of the input fuzzy sets are combined with output fuzzy sets using the extended sup-star composition, and the multiple rules are combined using the join operation [8]. However, the computing load involved in deriving the system output from a general type-2 FLS model is high in practice, and the general practice is to use the interval type-2 FLS in which the fuzzy sets F_i^l and G^l are interval fuzzy sets through which the computing of type-2 FLS can be greatly simplified. The membership grades of interval fuzzy sets can be fully characterized by their lower and upper membership grades of the footprint of uncertainty (FOU) separately [8].

Without the loss of generality, let $\mu_{F_i^l}(x) = [\underline{\mu}_{F_i^l}(x), \overline{\mu}_{F_i^l}(x)]$ and $\mu_{G^l}(y) = [\underline{\mu}_{G^l}(y), \overline{\mu}_{G^l}(y)]$ for each sample (x, y). The firing strength of interval type-2 FLS $\mu_{F^l}(x) = \bigcap_{i=1}^{n} \mu_{F_i^l}(x)$ is an interval [8], that is, $\mu_{F^l}(X) = [\underline{f}^i(X), \overline{f}^i(X)]$. In the proposed interval type-2 FLS, the meet operation under product t-norm is used, so that the firing strength is an interval type-1 set [8] as shown below:

$$f^i(X) = \left[\underline{f}^i(X), \overline{f}^i(X)\right] = \left[\underline{f}^i, \overline{f}^i\right], \tag{3}$$

where $\underline{f}^i(X)$ and $\overline{f}^i(X)$ can be rewritten as follows with $*$ representing the t-norm product operation:

$$\begin{aligned} \underline{f}^i(X) &= \underline{\mu}_{\overline{F}_1^j}(x_1) * \cdots * \underline{\mu}_{\overline{F}_p^j}(x_n), \\ \overline{f}^i(X) &= \overline{\mu}_{\overline{F}_1^j}(x_1) * \cdots * \overline{\mu}_{\overline{F}_p^j}(x_n). \end{aligned} \tag{4}$$

3.1.2. Type Reduction. The results from the inference engine are type-2 fuzzy sets. There is then the need to reduce the type-2 fuzzy sets to type-1 fuzzy sets in order to give room for defuzzification so that the final crisp outputs can be generated. Centre-of-sets (COS) type-reducer algorithm developed by Mendel [8] and Karnik et al. [62] has been used in this study because it provides reasonable computational complexity compared to others like the expensive centroid type reducer, though other types can still be investigated when the need arises as the research progresses. COS type reducer uses two steps in reducing the type-2 fuzzy sets as follows: (i) calculating the centroids of type-2 fuzzy rule consequences and (ii) calculating the reduced fuzzy sets. These stages are described in the following two subsections.

Computing the Centroids of Type-2 Fuzzy Rule Consequences. Suppose that the output of an interval type-2 FLS is represented by type-2 fuzzy sets G^t, where $t = 1, \ldots, T$. T is the number of output fuzzy sets. In this first stage, the centroids of all the T output fuzzy sets are calculated and they will be used in calculating the reduced sets in the next stage. The centroid of the ith output fuzzy set y^t is a type-1 interval set which can be expressed in the following equation [8, 62]:

$$y^t = \left[y_l^t, y_r^t\right] = \int_{\theta_1 \in J_{y1}} \int_{\theta_z \in J_{yz}} \frac{1}{\sum_{z=1}^{Z} y_z\theta_z / \sum_{z=1}^{Z} y_z\theta_z}, \tag{5}$$

where y_l^t and y_r^t are the leftmost and rightmost point of y^t, respectively.

Algorithm 1 is the iterative procedures developed by Mendel [8] Karnik et al. [62] to calculate the rightmost point y_r^t for each type-2 output fuzzy set, where Z is representing the number of discretised points for each output fuzzy set, $J_{y_z} = [L_z, R_z]$, $h_z = [h_z + R_z]/2$, $\Delta_z = [R_z - L_z]/2$, $z = 1, \ldots, Z$. Figure 2 demonstrates how to calculate h_z, L_z, R_z, and Δ_z needed by the Algorithm 1. The leftmost point y_l^t can be calculated in the similar way except at step 4 of Algorithm 1 where we set $\theta_z = h_z + \Delta_z$ when $z \le e$ and $\theta_z = h_z - \Delta_z$ when $z > e + 1$. This iterative procedure has been proven to converge in at most Z iterations to find y_r^t or y_l^t [8].

Computing the Reduced Fuzzy Sets. To calculate the type-reduced set, it is sufficient to compute its upper and lower bounds of the reduced set y_l and y_r, which can be expressed as follows:

$$y_l = \frac{\sum_{i=1}^{M} f_l^i y_l^i}{\sum_{i=1}^{M} f_l^i}, \qquad y_r = \frac{\sum_{i=1}^{M} f_r^i y_r^i}{\sum_{i=1}^{M} f_r^i}, \tag{6}$$

where f_l^i and y_l^i are the firing strength and the centroid of the output fuzzy set of ith rule $(i = 1, \ldots, M)$ associated with y_l, respectively. Similarly, f_r^i and y_r^i are the firing strength and the centroid of the output fuzzy set of ith rule $(i = 1, \ldots, M)$ associated with y_r, respectively.

Meanwhile, y_r can be calculated using the iterative Algorithm 2 as proposed in [5, 8]. Similarly, y_l can be calculated in the same way by setting $f_r^i = \overline{f}^i$ for $i \le R$

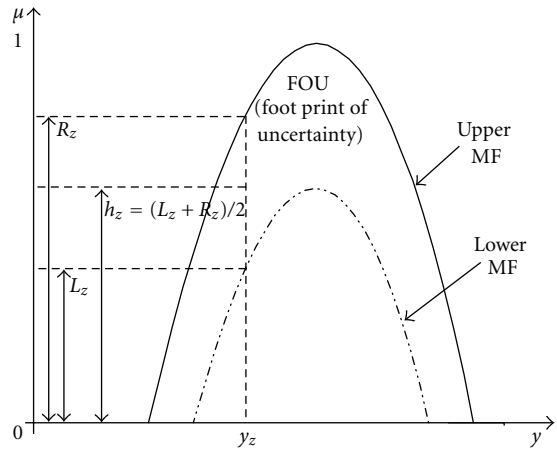


Figure 2: Diagrammatic representation of the computation of the parameters needed by each y_z in Algorithm 1 for computing y_r^t and y_l^t.

(1) Arrange the precalculated y_r^i from Figure 2 in an ascending order, that is, $f_r^1 \le f_r^2 \le \cdots \le f_r^M$
(2) Set $f_r^i = (\underline{f}_r^i + \overline{f}_r^i)/2$ for $i = 1, 2, \ldots M$, and calculate y_r using (6)
(3) Find $R \in [1, M-1]$ such that $y_r^R \le y_r' \le f_r^{R+1}$
(4) Set $f_r^i = \underline{f}_r^i$ for $i \le R$ and $f_r^i = \underline{f}_r^i + \overline{f}_r^i$ for $i > R$; and compute y_r'' using (6)
(5) Stop if $y_r'' = y_r'$, otherwise set $y_r' = y_r''$ and return to step 3

Algorithm 2: Iterative algorithm to compute y_r.

and $f_r^i = \underline{f}^i$ for $i > R$. The iterative procedure has also been proved to converge in no more than M iterations when computing either y_l or y_r [5].

3.1.3. Defuzzification. We defuzzify the type-reduced set to get a crisp output from the type-2 FLS. The final output of type-2 FLS is thus set to the average of y_l and y_r as shown below:

$$y(x) = \frac{y_l + y_r}{2}, \tag{7}$$

where $y(x)$ is the final crisp output.

3.1.4. The Steepest Descent Approach for Training FLS. The purpose of the training algorithm is to minimize the error function for E training epochs:

$$E(t) = \left(\frac{f\left(x^{(t)}\right) - y^{(t)}}{y^{(t)}} \right)^2, \quad t = 1, \ldots, N. \tag{8}$$

Consider an FLS with Gaussian membership functions, centre of sums type-reducer, average defuzzification, max-product composition, and product implication; it could be expressed by the equation

$$y\left(x^{(i)}\right) = f_s\left(x^{(i)}\right) = \frac{\sum_{l=1}^{M} \overline{y}^l \prod_{k=1}^{p} \exp\left[-\left(x_k^{(i)} - m_{F_k^l}\right)^2 / \left(2\sigma_{F_k^l}^2\right) \right]}{\sum_{l=1}^{M} \prod_{k=1}^{p} \exp\left[-\left(x_k^{(i)} - m_{F_k^l}\right)^2 / \left(2\sigma_{F_k^l}^2\right) \right]}, \quad i = 1, \ldots, N, \tag{9}$$

where: M is number of rules, p is number of antecedents, and N is number of data points; $m_{F_k^l}$ and $\sigma_{F_k^l}$ are the mean and standard deviation for the membership function, respectively.

Given an input-output training pair $(x^{(i)} : y^{(i)})$ also known as data point, we wish to design a fuzzy logic system (FLS) so that the error function is minimized. The steepest descent approach [8] can be applied to obtain the following recursions to update all the design parameters of this FLS in order to minimize the error function:

$$m_{F_k^l}(i+1) = m_{F_k^l}(i) - \alpha_m \left[f_s\left(x^{(i)}\right) - y^{(i)} \right] \left[\overline{y}^l(i) - f_s\left(x^{(i)}\right) \right] \times \frac{\left[x_k^{(i)} - m_{F_k^l}(i) \right]}{\sigma_{F_k^l}^2(i)} \varphi_l\left(x^{(i)}\right), \tag{10}$$

(1) Initialize the parameters of all the membership functions for all the rules, $m_{F_k^l}(0)$, $\overline{y}^l(0)$, and $\sigma_{F_k^l}(0)$.
(2) Set an end criterion to achieve convergence.
(3)**Repeat**
  (i) **for all** data points $(x^{(i)} : y^{(i)})$ only $i = 1, \ldots, N$
    (a) Propagate the next data point through the FLS.
    (b) Compute error.
    (c) Update the parameters of the membership functions using (10), (11), and (12).
  (ii) **end for** (** end for each input-output pair **)
  (iii) Compute the root mean square relative error (RMSRE) as in (13).
  (iv) Test the end criterion. If satisfied break.
Until (** end for each epoch **)

Algorithm 3: Back propagation algorithm for FLS.

$$\overline{y}^l(i+1) = \overline{y}^l(i) - \alpha_{\overline{y}}\left[f_s\left(x^{(i)}\right) - y^{(i)}\right]\varphi_l\left(x^{(i)}\right), \tag{11}$$

$$\begin{aligned}\sigma_{F_k^l}(i+1) &= \sigma_{F_k^l}(i) - \alpha_\sigma\left[f_s\left(x^{(i)}\right) - y^{(i)}\right]\left[\overline{y}^l(i) - f_s\left(x^{(i)}\right)\right] \\ &\quad \times \frac{\left[x_k^{(i)} - m_{F_k^l}(i)\right]^2}{\sigma_{F_k^l}^3(i)}\varphi_l\left(x^{(i)}\right).\end{aligned} \tag{12}$$

Now, the back propagation algorithm can be applied as in Algorithm 3 followed by RMSRE in (13):

$$\text{RMSRE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{\left[f_s\left(x^{(i)}\right) - y^{(i)}\right]^2}{y^{(i)}}}. \tag{13}$$

3.2. Sensitivity-Based Linear Learning Method (SBLLM). In [14], the authors proposed a new learning scheme in order to both speed up and avoid local minima convergence of the existing back propagation learning technique, while alleviating its other common weaknesses such as instability and inconsistency. This new learning strategy is called the sensitivity based linear learning method (SBLLM) scheme. It is a learning technique for two-layer feedforward neural networks based on sensitivity analysis, which uses a linear training algorithm for each of the two layers. First, random values are assigned to the outputs of the first layer; later, these initial values are updated based on sensitivity formulas, which use the weights in each of the layers and the process is repeated until convergence. Since these weights are learnt solving a linear system of equations, there is an important saving in computational time. The method also gives the local sensitivities of the least squared errors with respect to input and output data, with no extra computational cost, because the necessary information becomes available without extra calculations. This new scheme can also be used to provide an initial set of weights, which significantly improves the behaviour of other learning algorithms. The full theoretical basis for SBLLM and its performance has been demonstrated in [14], which contained its application to several learning problems examples in which it is compared with several learning algorithms and well known data sets. The results have shown a learning speed generally faster than other existing methods.

Sensitivity analysis is a very useful technique for deriving how and how much the solution to a given problem depends on data; see [15, 63, 64] and the references therein for more details. However, in [14] it was shown that sensitivity formulas can also be used as a novel supervised learning algorithm for two-layer feedforward neural networks that presents a high convergence speed.

Generally, SBLLM process is based on the use of the sensitivities of each layer's parameters with respect to its inputs and outputs and also on the use of independent systems of linear equations for each layer to obtain the optimal values of its parameters. In addition, it gives the sensitivities of the sum of squared errors with respect to the input and output data.

3.2.1. The Learning Process for the Sensitivity Based Linear Learning Method. Consider the two-layer feedforward neural network in Figure 2, where I is the number of inputs, J is the number of outputs, K is the number of hidden units, $x0s = 1$, $z0s = 1$, S is the number of data samples, and the superscripts (1) and (2) are used to refer to the first and second layer, respectively.

This network can be considered to be composed of two one-layer neural networks as is shown in Figure 3.

According to [14], considering the one-layer network in Figure 4, the set of equations relating inputs and outputs is given by

$$y_{js} = f_j\left(\sum_{i=0}^{I} w_{ji}x_{is}\right); \quad j = 1, 2, \ldots, J;\ s = 1, 2, \ldots, S. \tag{14}$$

where I is the number of inputs, J is the number of outputs, $x_{0s} = 1$, w_{ji} are the weights associated with neuron j, and S is the number of data points.

To learn the weight w_{ji}, the following sum of squared errors between the real and the desired output of the networks is usually minimized as:

$$P = \sum_{s=1}^{S}\sum_{j=1}^{J}\delta_{js}^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(y_{js} - f_j\left(\sum_{i=0}^{I} w_{ji}x_{is}\right)\right)^2. \tag{15}$$

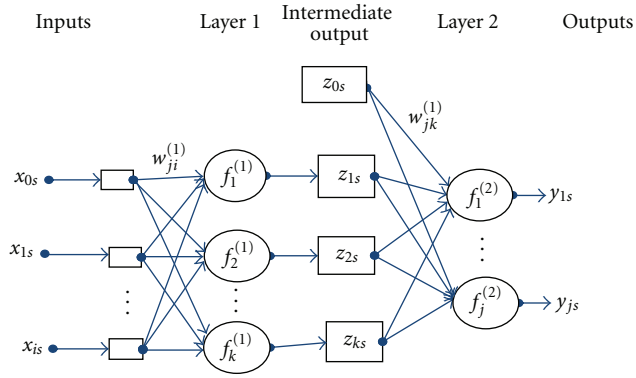


Figure 3: Two-layer feedforward neural network of the SBLLM.

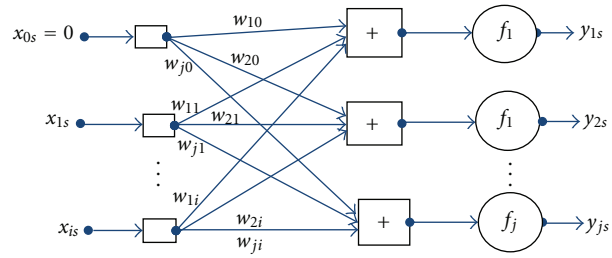


Figure 4: One-layer feedforward neural network combined to form the two-layer feedforward neural network of the SBLLM Scheme in Figure 2.

Assuming that the nonlinear activation functions, f_j, are invertible (which is the case for most of the commonly employed functions), otherwise, one can minimize the sum of squared errors before the nonlinear activation functions [14], that is,

$$Q = \sum_{s=1}^{S}\sum_{j=1}^{J}\varepsilon_{js}^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(\sum_{i=0}^{I}w_{ji}x_{is} - f_j^{-1}(y_{js})\right)^2, \tag{16}$$

which leads to the system of equations:

$$\begin{aligned}\frac{\partial Q}{\partial w_{jp}} &= 2\sum_{S=1}^{S}\left(\sum_{i=0}^{I}w_{ji}x_{is} - f_j^{-1}\left(y_{js}\right)\right)x_{ps}\\ &= 0; \quad p = 0, 1, \ldots, I; \ \forall j,\end{aligned} \tag{17}$$

that is,

$$\sum_{i=1}^{I}w_{ji}\sum_{s=1}^{S}x_{is}x_{ps} = \sum_{s=0}^{S}f_j^{-1}\left(y_{js}\right)x_{ps}; \quad p = 0, 1, \ldots, I; \ \forall j, \tag{18}$$

or

$$\sum_{i=0}^{I}A_{pi}w_{ji} = b_{pj}; \quad p = 0, 1, \ldots.., I; \ \forall j, \tag{19}$$

where

$$\begin{aligned}A_{pi} &= \sum_{s=1}^{S}x_{is}x_{ps}; \quad p = 0, 1, \ldots.., I; \ \forall i,\\ b_{pj} &= \sum_{s=1}^{S}f_j^{-1}\left(y_{js}\right)x_{ps}; \quad p = 0, 1, \ldots.., I; \ \forall j.\end{aligned} \tag{20}$$

Furthermore, for the one-layer neural network shown in Figure 4, according to [14, 15, 64], the sensitivities of the new cost function, Q, with respect to the input and output data can be obtained as:

$$\begin{aligned}\frac{\partial Q}{\partial y_{pq}} &= -\frac{2\left(\sum_{i=0}^{I}w_{pi}x_{iq} - f_p^{-1}\left(y_{pq}\right)\right)}{f_p'\left(y_{pq}\right)}; \quad \forall p, q,\\ \frac{\partial Q}{\partial x_{pq}} &= 2\sum_{j=1}^{J}\left(\sum_{i=0}^{I}w_{ji}x_{iq} - f_j^{-1}\left(y_{jq}\right)\right)w_{jp}; \quad \forall p, q.\end{aligned} \tag{21}$$

Based on [14], the learning method and the sensitivity formulas presented above can now be used to develop the SBLLM learning method for the two-layer feedforward networks shown in Figure 3. Now, consider the two-layer feedforward neural network in Figure 3, where I is the number of inputs, and include the four PVT properties independent variables namely solution gas-oil ratio, reservoir temperature, oil gravity, and gas relative density; J is the number of outputs, and it include the two target PVT properties (dependent variables) namely bubble point

pressure (P_b) and oil formation volume factor (B_{ob}); K is the number of hidden units, $x0s = 1$, $z0s = 1$; S is the number of data samples and the superscripts (1) and (2) are used to refer to the first and second layer, respectively.

Assuming that the intermediate layer outputs z are known, using (16), a new cost function for the two-layer feedforward neural network in Figure 3 is defined as:

$$\begin{aligned} Q(z) &= Q^{(1)}(z) + Q^{(2)}(z) \\ &= \sum_{s=1}^{S}\left[\sum_{k=1}^{K}\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks})\right)^2 \right. \\ &\qquad \left. + \sum_{j=1}^{J}\left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - f_k^{(2)^{-1}}\left(y_{js}\right)\right)^2\right]. \end{aligned} \tag{22}$$

Thus, using the outputs z_{ks} we can learn, for each layer independently, the weights $w_{ki}^{(1)}$ and $w_{jk}^{(2)}$ by solving the corresponding linear system of (19). After this, the sensitivity (see (21)) with respect to z_{ks} is calculated thus:

$$\begin{aligned} \frac{\partial Q}{\partial z_{ks}} &= \frac{\partial Q^{(1)}}{\partial z_{ks}} + \frac{\partial Q^{(2)}}{\partial z_{ks}} \\ &= -\frac{2\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks})\right)}{f_k^{\prime(1)}(z_{ks})} \\ &\quad + 2\sum_{j=1}^{J}\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - f_j^{(2)^{-1}}\left(y_{js}\right)\right) w_{jk}^{(2)}, \end{aligned} \tag{23}$$

with $k = 1, \ldots, K$, as $z_{0s} = 1$, for all s. After this, the values of the intermediate outputs z are modified using the Taylor series approximation:

$$Q(z + \Delta z) = Q(z) + \sum_{k=1}^{K}\sum_{s=1}^{S}\frac{\partial Q(z)}{\partial z_{ks}}\Delta z_{ks} \approx 0, \tag{24}$$

which leads to the following increments:

$$\Delta z = -\rho\frac{Q(z)}{\|\nabla Q\|^2}\nabla Q, \tag{25}$$

where ρ is a relaxation factor or step size.

3.2.2. Strengths and Weaknesses of SBLLM. (A) It has been established that the SBLLM offers an interesting combination of speed, reliability, and simplicity. In addition, based on the results obtained from the real-world experiments using the SBLLM learning algorithm, there are four main advantages of the SBLLM that can be summarized as follows, [14].

(i) *High speed in reaching the minimum error*: it was demonstrated in [14] that in all cases the SBLLM obtains its minimum mean squared error (MSE_{min}) just before the first four iterations and also sooner than the rest of the algorithms examined together. SBLLM gets its minimum error in an epoch for which the other algorithms are far from similar MSE values.

(ii) *Good performance*: it can be deduced not only that SBLLM stabilizes soon, but also the minimum MSE that it reaches is quite good and comparable to that obtained by the second order methods. Other methods compared to it never succeeded in attaining this minimum MSE (MSE_{min}) before the maximum number of epochs.

(iii) *Homogeneous behavior*: the SBLLM learning curve stabilizes easily and within short time. The SBLLM behaves homogeneously not only if we consider just the end of the learning process, but also during the whole process, in such a way that very similar learning curves were obtained for all iterations of different experiments. This is indicative of ability to handle prediction with high stability and consistency which are requisite of good prediction system particularly in oil and gas reservoir modeling.

(B) Weaknesses of SBLLM. One of the major weaknesses of this technique is its inability to model uncertainties which is a very important capability being sought for in today's predictive solutions particularly in oil and gas modeling where uncertainties are very common. Also the prediction accuracy of SBLLM is usually dependent on the nature of the problem at hand, that is, nature of the data set. Thus there is need to compliment its usage with other better model like type-2 FLS to achieve better performance, particularly in the face of uncertainties.

3.3. The Proposed T2-SBLLM Hybrid Framework. In this proposed hybrid system, type-2 fuzzy logic system is first used for modeling and uncertainty handling using its procedural components of fuzzification, inferencing, type-reduction, and defuzzification processes to generate a clean output, while the Sensitivity based linear learning method (SBLLM) is then trained using the cleaned output from type-2 FLS and then used to make the final predictions. This is then applied to the problem of PVT property prediction in the field of reservoir engineering, but it is extendable for use in other prediction or classification oriented applications.

3.3.1. Conceptual Design of the Proposed T2-SBLLM Hybrid Models. Hybrid computational intelligence is defined as any effective combination of intelligent techniques that performs superior or in a competitive way to simple standard intelligent techniques. The increased popularity of hybrid intelligent systems in recent times lies in the extensive success of these systems in many real-world complex problems [1]. Also, it is an established fact that every approach has its strengths and weaknesses; hence the need for hybrid models that is able to combine the strengths of the individual techniques while complementing the weaknesses of one method with the strength of the other.

In this section, smart hybridization of type-2 FLS and sensitivity based linear learning method (SBLLM) is investigated, in order to take advantages of hybrid systems in general while using each component to complement the other in order to take advantages of each framework in particular,

where possible, while alleviating the weaknesses of component members in order to achieve effective, stable, consistent, and accurate predictive solution. The methodology in this work is based on the standard Computational Intelligence approach to hybridization of ion different techniques. The hybrids are designed in order to benefit immensely from the strength of the individual techniques while complementing the weaknesses of each technique with the advantages of the others through smart and optimum combination of the cooperative and competitive characteristics of the individual techniques.

In this hybrid system, type-2 fuzzy logic system is first used for modeling and uncertainty handling using its procedural components of fuzzification, inferencing, type-reduction, and defuzzification processes to generate a clean output, while the sensitivity based linear learning method (SBLLM) is then trained using the cleaned output from type-2 FLS and then used to make the final predictions. This is then applied to the problem of PVT properties and permeability prediction in the field of reservoir engineering, but it is extendable for use in other prediction or classification oriented applications.

In this case, a T2-SBLLM hybrid system that comprises of two building blocks made up of type-2 fuzzy logic systems (T2) and sensitivity based linear learning method (SBLLM), has been built. Having divided the input data into training and testing sets using the stratified sampling approach, the training and testing sets are passed to the type-2 FLS block where all possible forms of uncertainties are adequately handled. This capability of type-2 FLS has been shown in several works such as in [10, 62, 65] and this has been confirmed in this work also. It is able to adequately handle different forms of uncertainties using its extension to a third dimension. The type-2 FLS undergoes the necessary processes of fuzzification, inferencing, type-reduction, and final defuzzification to generate final crisp output. The final outputs from the type-2 FLS for both training and testing set are then passed to the SBLLM block for training and testing purposes, respectively. Thus, the output from the type-2 training process is then used to train the SBLLM in readiness for prediction using the unseen test dataset that happens to be the output from the type-2 FLS testing process. It must be noted that the target values are never manipulated by the type-2 FLS block in order to avoid creating bias as a result of such process. This means that the actual values (also known as target values) are just made to be read only for type-2 FLS. Finally, the new unseen test data, which is the type-2 FLS testing output, is then passed to the trained SBLLM model to perform the final prediction task.

The role performed by the type-2 FLS block in this model is to ensure that dataset containing uncertainties has been properly handled and cleaned to generate output that is later passed to the SBLLM block for training and then final prediction purposes. As a result of this, it ensures that, during the training process, clean and trusted data is allowed to enter the SBLLM block that is responsible for carrying out the prediction task after the training process, and this in turn facilitates better performance of the SBLLM model produced. Figure 5 shows the conceptual design framework of the proposed T2-SBLLM hybrid system.

With the designed framework shown in Figure 5, the hybridization of type-2 FLS and sensitivity based linear learning method become clear to develop and implement. Meanwhile each of the designated steps in Figure 5 is briefly explained as follows.

Step 1. This is the step where the necessary real industrial dataset are made ready in preparation to be sent into the hybrid model.

Step 2. In this step, the available data is passed to type-2 fuzzy logic systems as the first component of the hybrid scheme. Several major subactivities take place here. These include fuzzification, type-2 FLS rule extraction, consequent matching, type reduction, and defuzzification processes. These processes ensure adequate uncertainty handling and modeling based on the entire processes of type-2 FLS. It must be noted that the dataset is already divided into training set and testing set and each is being processed separately, one after the other, at this stage. This means that, at the end of this step, two different outputs are generated, one from the training set and the other from the testing set. These two different outputs (which are the predicted target variable) are then presented to the next step of the hybrid scheme.

Step 3. In this step, the two different outputs of training and testing, coming from the type-2 FLS, are distinctly and, respectively, preserved in the form of training and testing sets to be passed to the next component of the hybrid model, which is SBLLM. Thus, this step separately passed the new training data set into the next step (Step 4) where training and calibration of the SBLLM take place, while the testing set is retained only to be passed to the Step 5 where testing of the trained SBLLM takes place.

Step 4. In this stage, SBLLM is being trained using the training output from type-2 FLS that has undergone adequate uncertainty handling procedures of type-2 FLS. Therefore, SBLLM is being trained using clean and better output from type-2 FLS. This ensures that SBLLM is being trained in such a way to prevent the effect of uncertainties on its performance, thereby facilitating the generation of better and improved model with the ultimate goal of achieving better performance accuracy.

Step 5. In this step, the trained SBLLM in previous step is being tested using the testing dataset retained in Step 3 for onward transfer to this present step. Thus, the SBLLM is being tested using the output from the type-2 FLS testing section. This ensures that clean and better data is passed to the SBLLM for final testing and the ultimate prediction needed. The testing at this step is the final testing for the entire hybrid model presented.

Step 6. This is the stage where final predicted output from the hybrid model is collected and necessary performance measure analysis and computations are carried out to

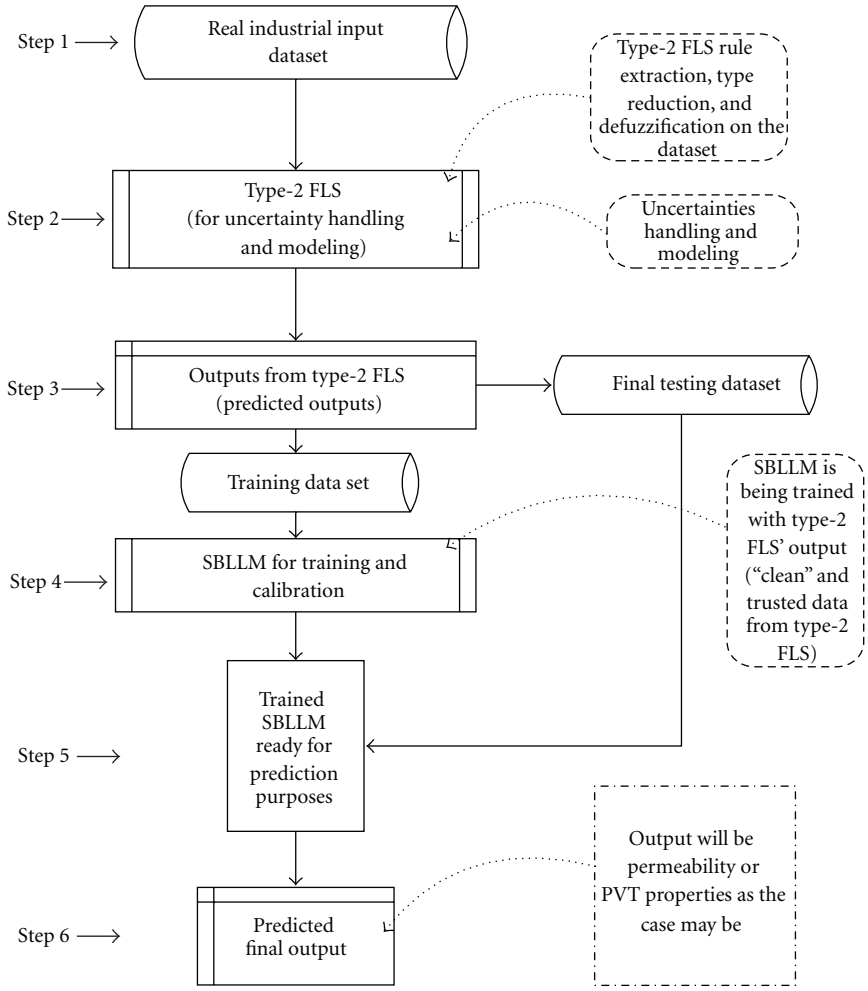


Figure 5: Schematic design framework for the proposed T2-SBLLM hybrid model for predicting PVT properties.

identify the performance accuracy gains of the proposed hybrid model over the individual constituting models. The most common statistical quality measures that are utilized in both data mining and petroleum engineering journals were employed in this stage, and their description is presented in Section 4.2.

To further make the flow of data through the entire processes clearer, a data flow diagram for the proposed T2-SBLLM hybrid model is presented as in Figure 6.

It must be stated here that the proposed model followed strictly the standard training and testing procedures in which the testing set is kept away as unseen data before it is sent to the model for testing. From the data flow diagram depicted in Figure 6, it is made clear how the dataset is first partitioned into training set and testing set. The standard procedure for the sequential implementation of hybrid model is followed in this work, where the preceding method (T2FLS) always acted on the dataset to generate an output that is then fed to the next model in the hybrid setup (i.e., SBLLM in this present work.). The well log attributes are first sent to the T2FLS for training. Based on the training set, T2FLS is made to generate an output representing its estimated permeability values. These estimated permeability values produced from the training section of T2FLS are then passed into the SBLLM as input to be used for training. Having trained the SBLLM model with the estimated permeability from T2FLS the system proceeded to the testing process as follows.

As shown in Figure 6, the testing set is separated into the attributes and the target (actual) permeability. The testing set attributes are first sent into T2FLS for testing and the T2FLS generates its final estimates of the target output (PVT properties). The estimated PVT property values produced from the testing section of T2FLS are then passed into the trained SBLLM as input to be used in carrying out its testing procedure. At the end of this SBLLM testing, final predicted values for the PVT properties are then generated as the final output from the hybrid scheme. These predicted outputs are then compared to the preserved target (actual) PVT properties in order to determine the accuracy of the final predicted output.

Depicted in Figure 7 is the network diagram demonstrating how the estimated values of PVT properties from T2FLS are made to enter the SBLLM network as input to the system.

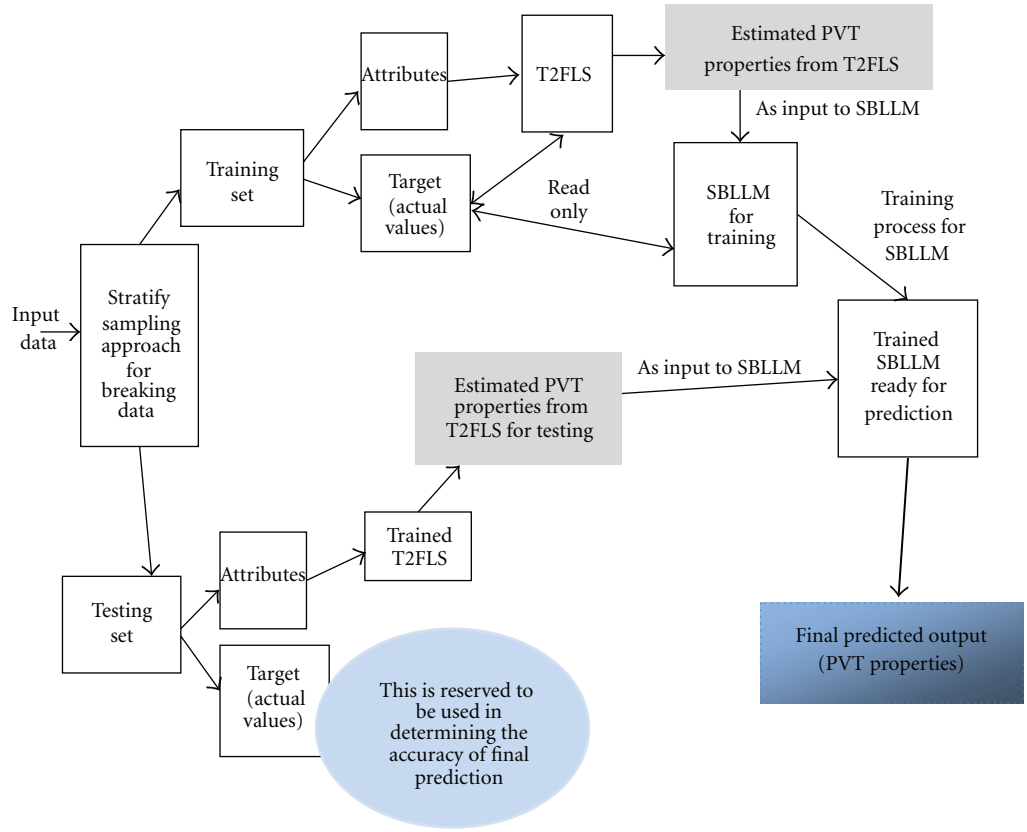


Figure 6: Data flow diagram of the proposed T2-SBLLM hybrid model.

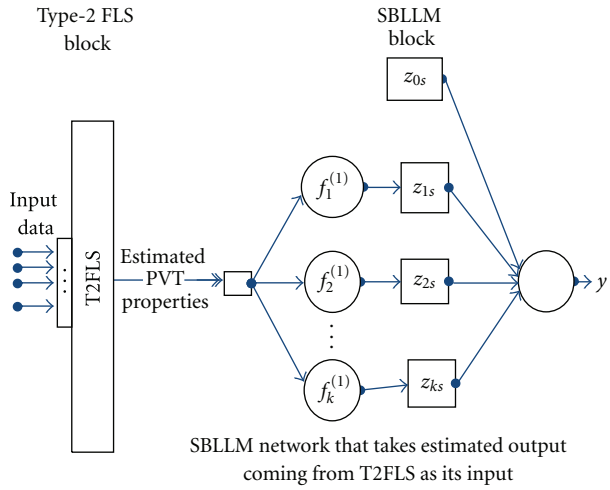


Figure 7: Network diagram of the proposed hybrid showing the flow of data from T2FLS into SBLLM for both training and testing phases (y stands for the final PVT properties output from the hybrid model).

It must be noted that the network diagram depicted in Figure 7 holds for both training and testing phases. For the testing phase, the training data set is fed into the T2FLS and the flow continues as depicted in the figure. While in case of testing phase, the T2FLS is fed with testing data set and the estimated permeability output from T2FLS then goes into the SBLLM as its input as shown in the figures.

It must however be noted that the training set is different from the testing set that is set aside for ascertaining the predictive capability of the proposed models. Also, it must be noted that the T2FLS is first used on both training and testing set one after the other before the estimated output from each case is then passed into the SBLLM block for training and testing, respectively, as detailed in Figures 6 and 7.

3.3.2. Optimal Parameters Search Procedure for SBLLM. The parameters associated with the SBLLM were optimized through a test-set-cross-validation on the available data set.

The details of the test-set-cross-validation for optimizing the SBLLM parameters goes thus: For each run of generated training and testing set, the values of RMSE and correlation coefficient were monitored for a group of parameters that includes a number of hidden neurons (N) and the activation functions (AF). Searching through all possible values of the parameters in a given range will identify the best performance measures and the corresponding values of the parameters for the fixed set of features. In our experiment, this process was repeated for every SBLLM activation function available, each time with an incremental step of parameters. The optimal values of the parameters and the kernel option associated with the best performance measure were identified. A summary of the procedure is as follows.

Step 1. Choose the initial "activation function" option from the list of available options.

Step 2. Identify the best values of the number hidden neuron through a test-set-cross-validation and store the corresponding performance measures.

Step 3. If there is no activation function option left, then go to Step 4. Otherwise, add the next activation function option and go to Step 2.

Step 4. Identify the best performance measure and its associated parameters values.

Step 5. Use the optimized activation function option and the parameters values to train the final SBLLM

Step 6. Calculate the performance measures for both the training and testing sets using the system obtained in the previous Step 5.

This is presented in mathematical form as follows.

Let the set A contain all the possible activation function options, the element of A is of the shape $A_i(j)$, where i is the activation function number, j is the selected number of hidden neurons, nf is the total number of activation functions available, and nh is the maximum number of hidden neuron assumed. Also pm represents performance measure taken, ix represents index for best activation function, and jx represents index for best number of layer.

The algorithm then goes thus:

initialization; $jx = 0$, $vx = 0$, $ix = 0$,

for $i = 1 \rightarrow nf$,

for $j = 1$ $j = 1 \rightarrow nh$,

pm $= f(A_i(j))$ {performance measure for the present parameters combination},

if pm is better than vx then $vx =$ pm,

$ix = i$; $jx = j$ {storing the index of the better parameter},

end,

end.

4. Empirical Study, Results, and Discussion

In order to carry out an empirical study, three distinct databases were acquired. To evaluate the performance of each modelling scheme, the entire database was divided, using the stratified sampling approach, into training set and testing set. The training set (70% of the entire dataset) was used for training and building the proposed implemented models (internal validation) while the testing set (remaining 30%) was used for testing and validating the models. For testing and evaluation of the newly proposed hybrid framework and to carry out effective comparative studies, the most common statistical quality measures that are utilized in both data mining and petroleum engineering journals were employed in this study, and their brief descriptions are given shortly.

4.1. Acquired Datasets. For this study, three distinct datasets have been acquired. The complete databases were earlier utilized in distinct published research articles. They include: (a) 160 observations-database; (b) 283 observations-database; and (c) 782 observations-database. Details of each are as follows.

(a) 160-Dataset This first database was drawn from the article [66] containing published correlations for estimating bubble point pressure and oil formation volume factor for Middle Eastern oils. This database contains 160 observation data drawn from Middle Eastern reservoirs.

(b) 283-Dataset This second database was contained in the works of [19, 53]. This database has 283 data points collected from different Saudi fields to predict the bubble point pressure and the oil formation volume factor at the bubble-point pressure for Saudi crude oils.

(c) 782-Dataset This third database was obtained from the works of [20, 50], This database contains 782 observations after removing the redundant 21 observations from the actual 803 data points. This data set was gathered from Malaysia, the Middle East, the Gulf of Mexico, and Colombia.

One of the unique attributes of the three databases is that they all share the same input attributes (independent variables) and these include gas-oil ratio, API oil gravity, relative gas density, and reservoir temperature.

4.1.1. Uncertainties and Complexity Involved in Oil and Gas Reservoir Data. The complexity involved in reservoir exploration, which invariably leads to different forms of uncertainties, randomness, or irregularity in data are hereby discussed, based on literatures and opinion of experts in the field of reservoir engineering. It is an established fact that reservoir characterization involves handling uncertainties [13]. For instance, there is no causal, mathematically describable relationship between the porosity and permeability of sedimentary rocks. While, at least theoretically, porosity is independent of grain size, permeability is strongly dependent on it, through the specific surface factor figuring in the

Kozeny-Carman equation [67]. Because of the lack of a well-defined porosity-permeability correlation, in permeability prediction from well logs other rock-physical properties are utilized, such as electrical, radioactive, and sonic properties of the rock, obtained from well logs. As the underlying physical principles connecting these very different, and generally indirectly measured quantities, and relating them to permeability, are not known as yet, the only way to proceed seems to be to rely on probabilistic techniques in one form or other and apply multivariable regression analysis, fuzzy algorithms, or artificial neural-network techniques (Professor Gabor Korvin, personal communication, October 17, 2010).

The problem of permeability prediction is especially complicated in carbonate rocks whose depositional and diagenetic history can be very complex, so that their permeability cannot be causally upscaled from core scale to reservoir scale, or even up to a few feet scale seen by the well logs. Larger than core-sized, vuggy, or fractured intervals in carbonates can result in permeability which at the scale of a few feet are significantly higher than the matrix permeability measured in core plugs. Swarms of fractures, if connected, yield very large flow rates, if disconnected, very low flow rates—and this capricious variability is not recorded (or deeply hidden) by the core permeability data (Professor Gabor Korvin, personal communication, October 17, 2010).

A further problem arising when matching core data and well-log data is that the depth value indicated on the well log is never more precise than one part in a thousand, that is an exact depth correspondence between core and well-log value is impossible [13, 68].

The rock properties, k, otherwise called permeability can be determined by time-based recording of only one variable, the pressure change in each reservoir. Experimental error during data acquisition propagates through the data reduction process leading to uncertainty in experimental results. In addition, unlike steady-state systems, the pressure-time curves are influenced by the compressive storage of the reservoirs and both the dimensions and properties of the sample [69]. Thus, uncertainty in permeability and PVT properties may arise from errors in measurement of sample dimension, fluid pressure, or reservoir storages. Since reservoirs are typically small for tight rock samples and irregular in shape due to the combination of tubing, fittings, valves, and pressure transducers, thus the uncertainty is expected to be higher (Prof. Gabor Korvin, personal communication, August 23, 2009).

4.2. Criteria for Performance Evaluation. Different quality measures can be used to judge the performance and accuracy of the models. This is done by carrying out statistical error analysis. To evaluate and compare the performance and accuracy of the proposed SBLLM models with the earlier mentioned standard neural networks and the three common empirical correlations in literatures, the most common statistical quality measures that are utilized in both petroleum engineering and data mining journals, namely, average absolute percent relative error (E_a), standard deviation (SD) and correlation coefficient (R^2) have been employed; see [48, 50] for details regarding their mathematical formulae. However, their brief descriptions are given below.

(i) Correlation Coefficient . The correlation coefficient measures the statistical correlation between the predicted and actual values. This method is unique in model evaluations. A higher number means a better model, with a "1" meaning perfect statistical correlation and a "0" meaning there is no correlation, indicating a failed performance. The formula is

$$\frac{\sum (y_a - y'_a)\left(y_p - y'_p\right)}{\sqrt{\sum (y_a - y'_a)^2 \sum \left(y_p - y'_p\right)^2}}, \tag{26}$$

where y_a and y_p are the actual and predicted values while y'_a and y'_p are the mean of the actual and predicted values.

(ii) Average Absolute Percent Relative Error (E_a). The relative error is the absolute error divided by the magnitude of the exact value. The percent error is the relative error expressed in terms of per 100. And thus, the average absolute percent relative error (E_a) is the average of the absolute percent relative error for all the cases. Mathematically put as:

$$E_a = \sum_{i=1}^{n} \left| \frac{y_a^i - y_p^i}{y_a^i} \times 100 \right|, \tag{27}$$

where n is the total number of samples, y_a is the original (actual) values known, and y_p is the predicted values

(iii) Standard Deviation (SD). Standard deviation is a measure of the average distance between individual data points and their mean. It is a measure of how stable a model result is when repeated over several runs. It provided a measure of confidence; the higher the standard deviation ("sigma"), the lower the prediction reliability. This is very useful for understanding the risk of data extrapolation. A model will be adjudged to be consistent and stable if the standard deviation is very low.

The mathematical formula for determining standard deviation is:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(X_i - \overline{X}\right)^2}. \tag{28}$$

An equivalent version of this formula is:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(X_i - \overline{X}\right)^2}{n}}. \tag{29}$$

This can be interpreted as the square root of a summation divided by n, where n is the total number of samples, X_i is ith sample, and $\overline{X}$ is the mean.

Percentage of Improvement. This is a quality measure devised to make clear and understandable, the amount of improvement a particular model had over another model. It can be

a percentage increase as in the case of correlation coefficient (R^2) or percentage decrease for the case of standard deviation (SD). For instance, for a model to be adjudged as better than another in terms of R^2, it must achieve a higher R^2 but in terms of SD, it has to achieve a lower value. Based on these two possible cases, to which any of the quality measures must belong to either of the two but not both, the formulae for calculating the percentage increase and decrease, respectively, are formulated as follows:

$$\text{percentage increase} = \frac{\text{difference between the two values}}{\text{the lower of the two values}} \times 100,$$

$$\text{percentage decrease} = \frac{\text{difference between the two values}}{\text{the higher of the two values}} \times 100. \tag{30}$$

4.3. Experimental Environments and Settings. To evaluate performance of the proposed T2-SBLLM hybrid modeling scheme, the acquired database is divided, using the stratified sampling approach, into 80% training set and 20% testing set for estimating how the investigated model performed on new unseen data. For testing and evaluation of the newly developed framework, and to carry out effective comparative studies viz-a-viz other earlier methods, the most common statistical quality measures that are utilized in both petroleum engineering and data mining journals were employed in this study, and they were already discussed in the preceding section. The training set is first pass through type-2 FLS block for proper uncertainty handling and then the output coming from the type-2 FLS is used to train the SBLLM model. Thereafter, testing data is then used to evaluate the predictive capability of the trained SBLLM model. We repeat both internal and external validation processes foreach of the considered models. The obtained results are presented in tables that follow shortly.

The training sets were used to build the models while the testing sets were utilized in evaluating the predictive capability of the models. As for the implementation, we did not use any ready-made software, the entire coding has been done using MATLAB though some MATLAB inbuilt functions, most especially in the case of SBLLM, and few others made available online, have been called and used in some cases. Also part of the type-2 fuzzy logic functions made available in [8] were also made use of.

In the case of the type-2 FLS based model, the implementation process proceeded by supplying the system with the available input data sets, one sample at a time, and the rules and membership functions are automatically leaned from the available input data. Gaussian membership function has been used based on two different learning criteria that include least squares and back-propagation. The same combination was utilized in training FLS membership function parameters. Further details on initializing, training, and validating type type-2 FLS have been presented in Section 3, and additional details could be found in [8, 70, 71].

Table 1: Testing results for 160-dataset when predicting B_{ob} and P_b. R^2: correlation coefficient, SD: standard deviation, E_a: average absolute percent relative error (AAPRE), P_b: bubble-point pressure, B_{ob}: oil formation volume factor.

Prediction methods	B_{ob}			P_b		
	R^2	SD	E_a	R^2	SD	E_a
Type-2 fuzzy model	0.994	0.09166	1.493	0.931	2.461	20.65
SBLLM	0.995	0.09154	1.200	0.961	4.060	35.54
T2-SBLLM hybrid	0.996	0.01427	1.013	0.986	2.134	17.65

As for the sensitivity based linear learning method (SBLLM) implementation, the number of hidden neuron was set to be 1000 while the activation function used was sigmoidal (sig) activation function. All these were arrived at using the optimisation procedure described earlier.

5. Results and Discussions

The results of comparisons using external validation checks (testing on unseen data) have been summarized in Tables 1, 2, and 3. The performance results from hybridized type-2-SBLLM model outperformed each of the constituting individual models, which is in line with the general establish fact, to date, that a hybrid scheme often performs better than the individual constituent parts. The proposed hybrid model showed high accuracy in predicting both PVT property values with a stable and accurate performance and achieved the lowest standard deviation in all cases, lowest absolute percent relative error, and the highest correlation coefficient in most cases in comparison to the individual constituent model. Detailed discussion of the results for each model follows shortly.

Judging from the results summarized in Tables 1 through 3, it is clear that the proposed hybrid scheme is better than individual methods because a good forecasting scheme should have the highest correlation coefficient (R^2), the lowest standard deviation (SD), and the lowest absolute percent relative error (E_a).

From the tables presented, it could be easily observed that the proposed hybrid system performs better than the two individual constituent models that include type-2 FLS and SBLLM. The hybrid has proven to be a better way to boost the performance of SBLLM as the results indicated that it has greatly improved upon the performance of SBLLM, up to 96.9% improvement in terms of standard deviation (SD), 8.6% improvement in terms of correlation coefficient (R^2), and finally up to 95% improvement in terms of average absolute percent relative error (E_a). The performance of the newly proposed hybrid greatly outperforms the standard SBLLM model thereby serving as a better improved form of SBLLM, while it has also performed better than the type-2 FLS. These results are in line with the established fact that hybrid models usually performed better than any of thier individual constituting models. Further result analyses are presented as follows.

It could be easily observed, for instance, in estimating bubble-point pressure (P_b) based on the 782 dataset, that

Table 2: Testing results for 283-dataset when predicting B_{ob} and P_b. R^2: correlation coefficient, SD: standard deviation, E_a: average absolute percent relative error (AAPRE), P_b: bubble-point pressure, B_{ob}: oil formation volume factor.

Prediction methods	B_{ob}			P_b		
	R^2	SD	E_a	R^2	SD	E_a
Type-2 fuzzy model	0.9948	0.2662	1.069	0.9978	0.8873	1.034721
SBLLM	0.9959	0.0372	0.852	0.9597	0.1427	30.205
T2-SBLLM hybrid	0.9978	0.00815	0.510	0.9968	0.0943	2.1034

Table 3: Testing results for 782-dataset when predicting B_{ob} and P_b. R^2: correlation coefficient, SD: standard deviation, E_a: average absolute percent relative error (AAPRE), P_b: bubble-point pressure, B_{ob}: oil formation volume factor.

Prediction methods	B_{ob}			P_b		
	R^2	SD	E_a	R^2	SD	E_a
Type-2 fuzzy model	0.9998	0.1625	0.1	0.9894	2.347	0.3432
SBLLM	0.9826	0.4069	2.742	0.9138	1.3328	38.6214
T2-SBLLM hybrid	0.9975	0.0421	1.9866	0.9927	1.031	3.6273

T2-SBLL Mhybrid system had 8.6% improvement over that of sensitivity based linear learning method (SBLLM) and 0.33% improvement over that of type-2 FLS in terms of correlation coefficient (R^2). In terms of standard deviation (SD), T2-SBLLM hybrid model had 22.7% improvement over SBLLM and 56.1% over type-2 FLS, while in terms of absolute percent relative error (E_a), T2-SBLLM hybrid model had 90.6% improvement over SBLLM and 90.5% over type-2 FLS. Similarly, for the case involving estimating oil formation volume factor (B_{ob}) using the 783-dataset, T2-SBLLM hybrid system had 1.5% improvement over that of sensitivity based linear learning method (SBLLM) while type-2 FLS had 0.2% improvement over it, in terms of correlation coefficient (R^2). In terms of standard deviation (SD), T2-SBLLM hybrid model had 89.7% improvement over SBLLM and 74.1% over type-2 FLS, while in terms of absolute percent relative error (E_a), T2-SBLLM hybrid model had 27.6% improvement over SBLLM. Other reported results also follow similar trends with T2-SBLLM hybrid model taking the lead always.

Moreover, for the case involving estimating oil formation volume factor (B_{ob}) based on the 283-dataset, T2-SBLLM hybrid model had 78.1% improvement over that of sensitivity based linear learning method (SBLLM) and 96.9% improvement over that of type-2 FLS, in terms of standard deviation (SD). In terms of absolute percent relative error (E_a), T2-SBLLM hybrid model had 40.1% improvement over SBLLM and 52.3% over type-2 FLS, while in terms of correlation coefficient (R^2), T2-SBLLM hybrid model had 0.2% improvement over SBLLM and 0.3% over type-2 FLS. Meanwhile, for the case involving estimating bubble-point pressure (P_b) using the 283-dataset, T2-SBLLM hybrid system had 3.9% improvement over that of sensitivity based linear learning method (SBLLM) in terms of correlation coefficient (R^2), 33.9% in terms of standard deviation (SD) and 90% in term of absolute percent relative error (E_a).

As for the case involving estimating bubble-point pressure (P_b) based on the 160-dataset, T2-SBLLM hybrid model had 47.4% improvement over that of sensitivity based linear learning method (SBLLM) and 13.3% improvement over that of type-2 FLS in terms of standard deviation (SD). In terms of absolute percent relative error (E_a), T2-SBLLM hybrid model had 50.3% improvement over SBLLM and 14.5% over type-2 FLS, while in terms of correlation coefficient (R^2), T2-SBLLM hybrid model had 2.6% improvement over SBLLM and 5.9% over type-2 FLS. Meanwhile, for the case involving estimating oil formation volume factor (B_{ob}) using the 160-dataset, T2-SBLLM hybrid system had 84.4% improvement over that of sensitivity based linear learning method (SBLLM) and 84.43% improvement over that of type-2 FLS in terms of standard deviation (SD). In terms of absolute percent relative error (E_a), T2-SBLLM hybrid model had 15.6% improvement over SBLLM and 32.2% over type-2 FLS, while in terms of correlation coefficient (R^2), T2-SBLLM hybrid model had 0.1% improvement over SBLLM and 0.2% over type-2 FLS in terms of correlation coefficient (R^2).

From the overall reported experimental results, it could be easily noted that T2-SBLLM hybrid model performed better in all fronts. This is evident as its quality measure values are consistently better than others in all fronts. Even though the performance of type-2 FLS and SBLLM might be closer on very few cases, there was no single case where they are closer in performance in terms of standard deviation (SD), which is a measure of the stability of the predictive systems. As for the SBLLM model, the newly proposed T2-SBLLM hybrid model outperformed it throughout the reported experimental results. This is an indication that the proposed approach has greatly improved the capability of the classical SBLLM through the incorporation of type-2 FLS as preprocessor to the actual SBLLM, in the form of what is popularly regarded as hybrid system. The overall results also indicate that the newly proposed T2-SBLLM hybrid model is able to consistently deal with the nature of reservoir data due to its ability to cater for all forms of uncertainties, using its type-2 FLS component while ensuring a better generalization and higher stability and consistency using its SBLLM component.

6. Conclusion and Recommendations

A new hybrid model combining type-2 fuzzy logic system (T2) and sensitivity based linear learning method (SBLLM) has been proposed and implemented. The proposed hybrid serves as a better improvement over the classical SBLLM by way of using type-2 FLS as precursor to the SBLLM model. The proposed T2-SBLLM hybrid has also been used to model the PVT properties of crude oil systems using three distinct published databases. This is to investigate the performance and accuracy of the proposed T2-SBLLM hybrid system, while at the same time solving the challenging prediction problems in oil and gas industry. Further conclusion emanating from this research and future work recommendation are presented as follows.

A new hybrid modelling scheme, through the appropriate combination of type-2 FLS and SBLLM, has been investigated, developed, and implemented, as predictive solution that takes care of all forms of uncertainties, while ensuring stable and consistent predictions. It has been shown, through adequate simulation works, that the newly proposed hybrid system is able to provide a better prediction for PVT properties of crude oil systems. Validation of the framework has been carried out using published databases. Indepth comparative studies have been carried out between this new framework and each of the constituent models that include type-2 FLS and SBLLM. Also the proposed method has been specially compared to the classical sensitivity based linear learning machines in order to show the improvement provided by the new hybrid model over SBLLM. The empirical results have confirmed the superiority of proposed T2-SBLLM hybrid over SBLLM in all fronts. Similar improvement patterns were also recorded against type-2 FLS though at a lesser degree of improvement. Thus, the overall empirical results from the experimental works and simulations show that the proposed model outperformed all the other individual constituting models in all fronts. Given any new data, the proposed T2-SBLLM will be able to handle uncertainties that might be present and perform the required prediction effectively with stable and consistent results through unique combination of uncertainty handling capability of type-2 FLS and the unique generalization capability, consistency and stability of sensitivity based linear learning methods.

The proposed system will also be useful for other classification problems, bearing in mind that regression model can easily be modified to take care of classification problems unlike the classification model that cannot easily be made to carry out regression. Thus this work should be seen as a cutting edge solution in the field of pattern recognition, in general, as a tool for both regression and classifications.

As an intake from the promising results from this work, it is suggested, as a form of future work, that this newly proposed system should be considered as viable tools for other reservoir engineering problems like that of PVT properties, porosity, history matching, lithofacies, and other reservoir engineering properties, while also looking into its usefulness in other germane fields such as time series forecasting, bioinformatics, intrusion detection systems, and others.

Acknowledgments

The Zamalah/Institutional Ph.D. scholarship provided by UTM and The Ministry of Higher Education of Malaysia is hereby acknowledged. King Fahd University of Petroleum and Minerals, Saudi Arabia (KFUPM) is hereby acknowledged for the use of some facilities in the course of this research work.

References

[1] A. Giovanni and L. Vincenzo, "Using FML and fuzzy technology in adaptive ambient intelligence environments," *International Journal of Computational Intelligence Research*, vol. 1, no. 2, pp. 171–182, 2005.

[2] N. N. Karnik and J. M. Mendel, "Type-2 fuzzy logic systems: type-reduction," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2046–2051, IEEE, October 1998.

[3] Y. Li, X. Sun, J. Hua, and C. Gong, "Modelling redundant structure in ecosystem by type-2 fuzzy logic system," *Ecological Modelling*, vol. 211, no. 1-2, pp. 113–120, 2008.

[4] Q. Liang and N. N. Karnik, "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 30, no. 3, pp. 329–339, 2000.

[5] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, 2000.

[6] Q. Liang and J. M. Mendel, "Overcoming time-varying co-channel interference using type-2 fuzzy adaptive filters," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 12, pp. 1419–1428, 2000.

[7] M. H. Fazel Zarandi, B. Rezaee, I. B. Turksen, and E. Neshat, "A type-2 fuzzy rule-based expert system model for stock price analysis," *Expert Systems with Applications*, vol. 36, no. 1, pp. 139–154, 2009.

[8] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, 1st edition, 2001.

[9] J. M. Mendel, "Fuzzy sets for words: a new beginning," in *Proceedings of the 12th IEEE International conference on Fuzzy Systems*, pp. 37–42, Los Angeles, Calif, USA, May 2003.

[10] D. Wu and J. M. Mendel, "A vector similarity measure for linguistic approximation: interval type-2 and type-1 fuzzy sets," *Information Sciences*, vol. 178, no. 2, pp. 381–402, 2008.

[11] M. R. Mashinchi and A. Selamat, "Constructing a customer's satisfactory evaluator system using GA-based fuzzy artificial neural networks," *World Applied Sciences Journal*, vol. 5, no. 4, pp. 432–440, 2008.

[12] M. Reza Mashinchi and A. Selamat, "An improvement on genetic-based learning method for fuzzy artificial neural networks," *Applied Soft Computing Journal*, vol. 9, no. 4, pp. 1208–1216, 2009.

[13] K. I. Ali, R. Mohammadreza, and A. M. Seyed, "A fuzzy logic approach for estimation of permeability and rock type from conventional well log data: an example from the Kangan reservoir in the Iran Offshore Gas Field," *Journal of Geophysics Engineering*, vol. 3, no. 4, pp. 356–369, 2006.

[14] E. Castillo, B. Guijarro-Berdiñas, O. Fontenla-Romero, and A. Alonso-Betanzos, "A very fast learning method for neural networks based on sensitivity analysis," *Journal of Machine Learning Research*, vol. 7, pp. 1159–1182, 2006.

[15] E. Castillo, A. Conejo, A. S. Hadi, and A. Fernández-Canteli, "A general method for local sensitivity analysis with application to regression models and other optimization problems," *Technometrics*, vol. 46, no. 4, pp. 430–444, 2004.

[16] O. Omole, O. A. Falode, and A. D. Deng, "Prediction of nigerian crude oil viscosity using artificial neural network," *International Journal for Petroleum Processing*, vol. 51, no. 3, pp. 181–188, 2009.

[17] E. A. Osman, M. A. Al-Marhoun, and K. Fand, "Artificial neural networks models for predicting PVT properties of oil field brines," in *Proceedings of the 14th SPE Middle East Oil and Gas Show and Conference (MEOS '05)*, pp. 1501–1516, Bahrain, March 2005.

[18] M.B. Standing, "Oil-system correlation," in *Petroleum Production Handbook*, T. C. Frick, Ed., McGraw-Hill, New York, NY, USA, 2nd edition, 1962.

[19] A. O. Kumoluyi and T. S. Daltaban, "Higher-order neural networks in petroleum engineering," in *Proceedings of the 64th Annual Western Regional Meeting*, pp. 555–570, Longbeach, Calif, USA, March 1994.

[20] H. M. Goda, "Prediction of the PVT data using neural network computing theory," in *Proceedings of the 27th Annual SPE International Technical Conférence and Exhibition in Abuja*, Abuja, Nigeria, 2003.

[21] E. A. El-Sebakhy, "Forecasting PVT properties of crude oil systems based on support vector machines modeling scheme," *Journal of Petroleum Science and Engineering*, vol. 64, no. 1–4, pp. 25–34, 2009.

[22] A.E. El-Sebakhy, A. Abdulraheem, M. Ahmed et al., "Functional network as a novel approach for prediction of permeability in a carbonate reservoir," in *Proceedings of the SPE Conference*, 2007.

[23] M. B. Standing, "A pressure-volume-temperature correlation for mixtures of california oils and gases," *Drilling and Production Practice*, pp. 275–287, 1947.

[24] O. Glaso, "Generalized pressure-volume temperature correlations," *Journal of Petroleum Technology*, vol. 32, pp. 785–795, 1980.

[25] M.A. Al-Marhoun, "New correlation for formation volume factor of oil and gas mixtures," *Journal of Canadian Petroleum Technology*, vol. 31, no. 3, pp. 2–22, 1992.

[26] R. Labedi, "Use of production data to estimate volume factor, density and compressibility of reservoir fluids," *Journal of Petroleum Science and Engineering*, vol. 4, no. 4, pp. 375–390, 1990.

[27] M. E. Dokla and M. E. Osman, "Correlation of PVT properties for UAE crudes," *SPE Formation Evaluation*, vol. 7, no. 1, pp. 41–46, 1992.

[28] H. Y. Al-Yousef and M. A. Al-Marhoun, "Discussion of correlation of PVT properties for United Arab Emirates crudes," in *Proceedings of the Society of Petroleum Engineers Conference*, UAE, 1993.

[29] S. M. Macary and M. H. El-Batanoney, "Derivation of PVT correlations for the gulf of Suez crude oils," in *Proceedings of the EGPC 11th Petroleum Exploration and Production Conference*, Cairo, Egypt, 1992.

[30] A. M. Saleh, I. S. Maggoub, and Y. Asaad, "Evaluation of empirically derived PVT properties for Egyptian oils," in *Proceedings of the Middle East Oil Show and Conferenc*, SPE, Bahrain, 1987.

[31] M. I. Omar and A. C. Todd, "Development of new modified black oil correlations for Malaysian crudes," in *Proceedings of the Asia Pacific Oil and Gas Conference*, pp. 211–219, Singapore, February 1993.

[32] G. E. Petrosky and F. F. Farshad, "Pressure-volume-temperature correlations for Gulf of Mexico crude oils," in *Proceedings of the SPE Annual Technical Conference and Exhibition*, pp. 395–406, Houston, Tex, USA, October 1993.

[33] T. Kartoatmodjo and Z. Schmidt, "Large data bank improves crude physical property correlations," *Oil and Gas Journal*, vol. 92, no. 27, pp. 51–55, 1994.

[34] R. A. Almehaideb, "Improved PVT correlations for UAE crude oils," in *Proceedings of the 10th Middle East Oil Show & Conference*, pp. 109–120, Bahrain, March 1997.

[35] W. D. McCain, "Reservoir-fluid property correlations—state of the art," *SPE Reservoir Engineering*, vol. 6, no. 2, pp. 266–272, 1991.

[36] A. M. Elsharkawy, A. A. Elgibaly, and A. A. Alikhan, "Assessment of the PVT correlations for predicting the properties of Kuwaiti crude oils," *Journal of Petroleum Science and Engineering*, vol. 13, no. 3-4, pp. 219–232, 1995.

[37] M. A. Mahmood and M. A. Al-Marhoun, "Evaluation of empirically derived PVT properties for Pakistani crude oils," *Journal of Petroleum Science and Engineering*, vol. 16, no. 4, pp. 275–290, 1996.

[38] H. H. Hanafy, S. M. Macary, Y. M. ElNady, A. A. Bayomi, and M. H. El Batanony, "Empirical PVT correlations applied to Egyptian crude oils exemplify significance of using regional correlations," in *Proceedings of the SPE International Symposium on Oilfield Chemistry*, pp. 733–737, Houston, Tex, USA, February 1997.

[39] A. A. Al-Shammasi, "Bubble point pressure and oil formation volume factor correlations," in *Proceedings of the 11th SPE Middle East Oil Show & Conference*, pp. 241–256, Bahrain, February 1999.

[40] A. A. Al-Shammasi, "A review of bubblepoint pressure and oil formation volume factor correlations," *SPE Reservoir Evaluation and Engineering*, vol. 4, no. 2, pp. 146–160, 2001.

[41] O. O. Bello, K. M. Reinicke, and P. A. Patil, "Comparison of the performance of empirical models used for the prediction of the PVT properties of crude oils of the niger delta," *Petroleum Science and Technology*, vol. 26, no. 5, pp. 593–609, 2008.

[42] S. S. Ikiensikimama and O. Ogboja, "New bubblepoint pressure empirical PVT correlation," in *Proceedings of the 33rd Annual SPE International Technical Conference and Exhibition*, Abuja, Nigeria, 2009.

[43] S. Mohaghegh, "Virtual-intelligence applications in petroleum engineering—part I—artificial neural networks," *Journal of Petroleum Technology*, vol. 52, no. 9, pp. 64–73, 2000.

[44] S. Mohaghegh and S. Ameri, "An artificial neural network as a valuable tool for petroleum engineers," in *Proceedings of the Society of Petroleum Engineers*, 1994.

[45] J. K. Ali, "Neural networks: a new tool for the petroleum industry," in *Proceedings of the European Petroleum Computer Conference*, Aberdeen, UK, 1994.

[46] A. R. Moghadassi, F. Parvizian, S. M. Hosseini, and A. R. Fazlali, "A new approach for estimation of PVT properties of pure gases based on artificial neural network model," *Brazilian Journal of Chemical Engineering*, vol. 26, no. 1, pp. 199–206, 2009.

[47] H. Hassanzadeh, M. Pooladi-Darvish, A. M. Elsharkawy, D. W. Keith, and Y. Leonenko, "Predicting PVT data for CO_2—brine mixtures for black-oil simulation of CO_2 geological storage,"

International Journal of Greenhouse Gas Control, vol. 2, no. 1, pp. 65–77, 2008.

[48] R. O. Duda, P. E. Hart, and D. G. Stock, *Pattern Classification*, John Wiley and Sons, New York, NY, USA, 2001.

[49] A. Sharifi, A. Moghadassi, F. Parvizian, and S. M. Hosseini, "Prediction of Pvt properties of ammonia by using artificial neural network and equations of state," *ARPN Journal of Engineering and Applied Sciences*, vol. 3, no. 6, pp. 18–27, 2008.

[50] E. A. Osman, O. A. Abdel-Wahhab, and M. A. Al-Marhoun, "Prediction of oil PVT properties using neural networks," in *Proceedings of the SPE Middle East Oil Show*, pp. 893–906, Bahrain, March 2001.

[51] R. B. Gharbi and A. M. Elsharkawy, "Neural network model for estimating the PVT properties of Middle East crude oils," in *Proceedings of the 10th Middle East Oil Show & Conference*, pp. 151–166, Bahrain, March 1997.

[52] N. Varotsis, V. Gaganis, J. Nighswander, and P. Guieze, "Novel non-iterative method for the prediction of the PVT behavior of reservoir fluids," in *Proceedings of the SPE Annual Technical Conference and Exhibition*, Houston, Tex, USA, October 1999.

[53] M. A. Al-Marhoun and E. A. Osman, "Using artificial neural networks to develop new PVT correlations for Saudi crude oils," in *Proceedings of the 10th Abu Dhabi International Petroleum Exhibition and Conference (ADIPEC '02)*, Abu Dhabi, UAE, 2002.

[54] A. Moghadassi, F. Parvizian, and S. Hosseini, "A new approach based on artificial neural networks for prediction of high pressure vapor-liquid equilibrium," *Australian Journal of Basic and Applied Sciences*, vol. 3, no. 3, pp. 1851–1862, 2009.

[55] A. M. Elsharkawy, "Modeling the properties of crude oil and gas systems using RBF network," in *Proceedings of the Asia Pacific Oil & Gas Conference*, pp. 35–46, Perth, Australia, October 1998.

[56] E. A. Osman and R. E. Abdel-Aal, "Abductive networks: a new modeling tool for the oil and gas industry," in *Proceedings SPE Asia Pacific Oil and Gas Conference and Exhibition*, pp. 487–493, Melbourne, Australia, October 2002.

[57] A. E. El-Sebakhy, A. Abdulraheem, M. Ahmed et al., "Functional network as a novel approach for prediction of permeability in a carbonate reservoir," in *SPE Conference*, 2007.

[58] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 117–127, 2002.

[59] E. Cox, "Adaptive fuzzy systems," *IEEE Spectrum*, vol. 30, no. 2, pp. 27–31, 1993.

[60] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, NY, USA, 1982.

[61] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[62] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999.

[63] E. Castillo, A. Cobo, J. Manuel Gutiérrez, and E. Pruneda, "Working with differential, functional and difference equations using functional networks," *Applied Mathematical Modelling*, vol. 23, no. 2, pp. 89–107, 1999.

[64] E. Castillo, J. M. Gutiérrez, and A. S. Hadi, "Sensitivity analysis in discrete Bayesian networks," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 27, no. 4, pp. 412–423, 1997.

[65] M. H. Fazel Zarandi, B. Rezaee, I. B. Turksen, and E. Neshat, "A type-2 fuzzy rule-based expert system model for stock price analysis," *Expert Systems with Applications*, vol. 36, no. 1, pp. 139–154, 2009.

[66] M. A. Al-Marhoun, "PVT correlations for Middle East crude oils," *Journal of Petroleum Technology*, vol. 40, no. 5, pp. 650–666, 1988.

[67] S. Mohaghegh, B. Balan, and S. Ameri, "State-of-the-art in permeability determination from well log data—part 2—verifiable, accurate permeability predictions, the touch-stone of all models," in *Proceedings of the 1995 Eastern Regional Conference*, pp. 43–47, Morgantown, WVa, USA, September 1995.

[68] N. Hurtado, M. Aldana, and J. Torres, "Comparison between neuro-fuzzy and fractal models for permeability prediction," *Computational Geosciences*, vol. 13, no. 2, pp. 181–186, 2009.

[69] I. Song, A. P. Rathbun, and D. M. Saffer, "Uncertainty of permeability and specific storage due to experimental error during data acquisition for pulse-transient technique," in *Proceedings of the American Geophysical Union Fall Meeting*, 2011.

[70] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 117–127, 2002.

[71] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999.

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 698062, 11 pages
doi:10.1155/2012/698062

Research Article

Speedup of Interval Type 2 Fuzzy Logic Systems Based on GPU for Robot Navigation

Long Thanh Ngo, Dzung Dinh Nguyen, Long The Pham, and Cuong Manh Luong

Department of Information Systems, Le Quy Don Technical University, No 100, Hoang Quoc Viet St., Cau Giay, Hanoi, Vietnam

Correspondence should be addressed to Long Thanh Ngo, ngotlong@gmail.com

Received 23 March 2012; Accepted 31 July 2012

Academic Editor: Oscar Montiel Ross

As the number of rules and sample rate for type 2 fuzzy logic systems (T2FLSs) increases, the speed of calculations becomes a problem. The T2FLS has a large membership value of inherent algorithmic parallelism that modern CPU architectures do not exploit. In the T2FLS, many rules and algorithms can be speedup on a graphics processing unit (GPU) as long as the majority of computation a various stages and components are not dependent on each other. This paper demonstrates how to install interval type 2 fuzzy logic systems (IT2-FLSs) on the GPU and experiments for obstacle avoidance behavior of robot navigation. GPU-based calculations are high-performance solution and free up the CPU. The experimental results show that the performance of the GPU is many times faster than CPU.

1. Introduction

Graphic processing units (GPUs) give a new way to perform general purpose computing on hardware that is better suited for the complicated fuzzy logic systems. However, the installation of these systems on the GPUs is also difficult because many algorithms are not designed in a parallel format conducive to GPU processing. In addition, there may be too many dependencies at various stages in the algorithm that will slow down GPU processing.

Type 2 fuzzy logic has been developed in theory and practice to obtain achievement for real applications [1–10]. A review of the methods used in the design of interval type 2 fuzzy controllers has been considered [11]. However, the complexity of T2FLS is still large and many researches focus to reduce these problems on the approach to algorithm or hardware implementation. Some proposals implement type 2 FLS focus on the design, and software development for coding a high-speed defuzzification stage based on the average method of two type 1 FLS [12] or the optimization of an incremental fuzzy PD controller based on a genetic algorithm [13]. More recent works, where an interval type 2 FIS Karnik-Mendel is designed, tested and implemented based on hardware implementation [14]. Using GPUs for general purpose computing is mentioned in many researches, recently, to speed up complicated algorithms by parallelizing to suitable GPU architecture, especially for applications of fuzzy logic. Anderson et al. [15] presented a GPU solution for the fuzzy C-means (FCMs). This solution used OpenGL and Cg to achieve approximately two orders of magnitude computational speedup for some clustering profiles using an nVIDIA 8800 GPU. They later generalized the system for the use of non-Euclidean metrics [16]. Further, Sejun Kim [17] describes the method used to adapt a multilayer trees structure composed of fuzzy adaptive units into CUDA platforms. Chiosa and Kolb [18] present a framework for mesh clustering solely implemented on the GPU with a new generic multilevel clustering technique. Chia et al. [19] proposes the implementation of a zero-order TSK-fuzzy neural network (FNN) on GPUs to reduce training time. Harvey et al. [20] present a GPU solution for fuzzy inference. Anderson et al. [21] present a parallel implementation of fuzzy inference on a GPU using CUDA. Again, over two orders of speed improvement of this

naturally parallel algorithm can be achieved under particular inference profiles. One problem with this system, as well as the FCM GPU implementation, is that they both rely upon OpenGL and Cg (graphics libraries), which makes the system and generalization of its difficult for newcomers to GPU programming.

Therefore, we carried out fuzzy logic systems analysis in order to take advantage of GPUs processing capabilities. The algorithm must be altered in order to be computed fast on a GPU. In this paper, we explore the use of nVIDIA's Compute Unified Device Architecture (CUDA) for the implementation of an interval type 2 fuzzy logic system (IT2FLS). This language exposes the functionality of the GPU in a language that most programmers are familiar with, the C/C++ language that the masses can understand and more easily integrate into applications that do not have the need otherwise to interface with a graphics API. Experiments are implemented for obstacle avoidance behavior of robot navigation based on nVIDIA platform with the summarized reports on runtime.

The paper is organized as follows: Section 2 presents an overview on GPUs and CUDA; Section 3 introduces the interval type 2 fuzzy logic systems; Section 4 proposes a speedup of IT2FLS using GPU and CUDA; Section 5 presents experimental results of IT2FLS be implemented on GPUs in comparing with on CPU; Section 6 is conclusion and future works.

2. Graphics Processor Units and CUDA

Traditionally, graphics operations, such as mathematical transformations between coordinate spaces, rasterization, and shading operations have been performed on the CPU. GPUs were invented in order to offload these specialized procedures to advanced hardware better suited for the task at hand. Because of the popularity of gaming, movies, and computer-aided design, these devices are advancing at an impressive rate. Classically, before the advent of CUDA, general purpose programming on a GPU (GPGPU) was performed by translating a computational procedure into a graphics format that could be executed in the standard graphics pipeline. This refers to the process of encoding data into a texture format, identifying sampling procedures to access this data, and converting the algorithms into a process that utilized rasterization (the mapping of array indices to graphics fragments) and frame buffer objects (FBO) for multipass rendering. GPUs are specialised stream processing devices.

This processing model takes batches of elements and computes a similar independent calculation in parallel to all elements. Each calculation is performed with respect to a program, typically called a kernel. GPUs are growing at a faster rate than CPUs, and their architecture and stream processing design makes them a natural choice for many algorithms, such as computational intelligence algorithms that can be parallelised.

nVIDIA's CUDA is a data-parallel computing environment that does not require the use of a graphics API, such as OpenGL and a shader language. CUDA applications are created using the C/C++ language. CPU and GPU programs are developed in the same environment (i.e., a single C/C++ program), and the GPU code is later translated from C/C++ to instructions to be executed by the GPU. nVIDIA has even gone as far as providing a CUDA Matlab plugin. A C/C++ program using CUDA can interface with one GPU or multiple GPUs can be identified and utilized in parallel, allowing for unprecedented processing power on a desktop or workstation.

CUDA allows multiple kernels to be run simultaneously on a single GPU. CUDA refers to each kernel as a grid. A grid is a collection of blocks. Each block runs the same kernel but is independent of each other (this has significance in terms of access to memory types). A block contains threads, which are the smallest divisible unit on a GPU. This architecture is shown in Figure 1.

The next critical component of a CUDA application is the memory model. There are multiple types of memory and each has different access times. The GPU is broken up into read-write perthread registers, read-write perthread local memory, read-write per-block shared memory, read-write per-grid global memory, read-only per-grid constant memory, and read-only per-grid texture memory. This model is shown in Figure 2.

Texture and constant memory have relatively small access latency times, while global memory has the largest access latency time. Applications should minimize the number of global memory reads and writes. This is typically achieved by having each thread read its data from global memory and store its content into shared memory (a block level memory structure with smaller access latency time than global memory). Threads in a block synchronize after this step. Memory is allocated on the GPU using a similar mechanism to malloc in C, using the functions cudaMalloc and cudaMallocArray. GPU functions that can be called by the host (the CPU) are prefixed with the symbol "*global*", GPU functions that can only be called by the GPU are prefixed with "*device*", and standard functions that are callable from the CPU and executed on the CPU are prefixed with "*host*" (or the symbol can be omitted, as it is the default). GPU functions can take parameters, as in C. When there are a few number of variables that the CPU would like to pass to the GPU, parameters are a good choice; otherwise, such as in the case of large arrays, the data should be stored in global, constant, or texture memory and a pointer to this memory is passed to the GPU function. Whenever possible, data should be kept on the GPU and not transferred back and forth to the CPU.

3. Interval Type 2 Fuzzy Logic Systems

3.1. Type 2 Fuzzy Sets. A type 2 fuzzy set in X is denoted $\widetilde{A}$, and its membership grade of $x \in X$ is $\mu_{\widetilde{A}}(x,u)$, $u \in J_x \subseteq [0,1]$, which is a type 1 fuzzy set in $[0,1]$. The elements of domain of $\mu_{\widetilde{A}}(x,u)$ are called primary memberships of x in $\widetilde{A}$, and memberships of primary memberships in $\mu_{\widetilde{A}}(x,u)$ are called secondary memberships of x in $\widetilde{A}$.

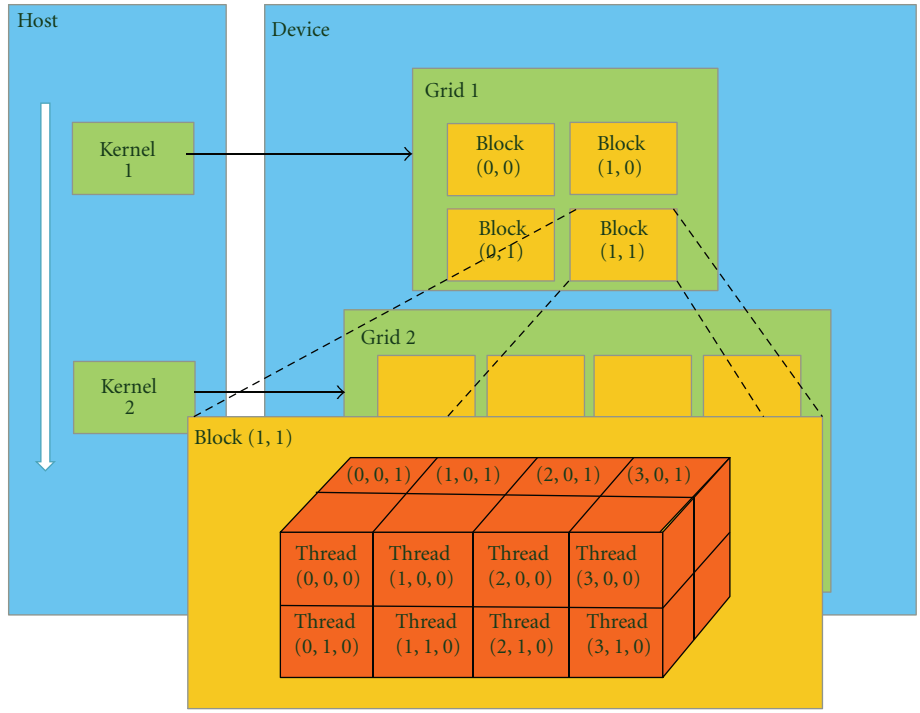


Figure 1: CUDA-processing model design [22].

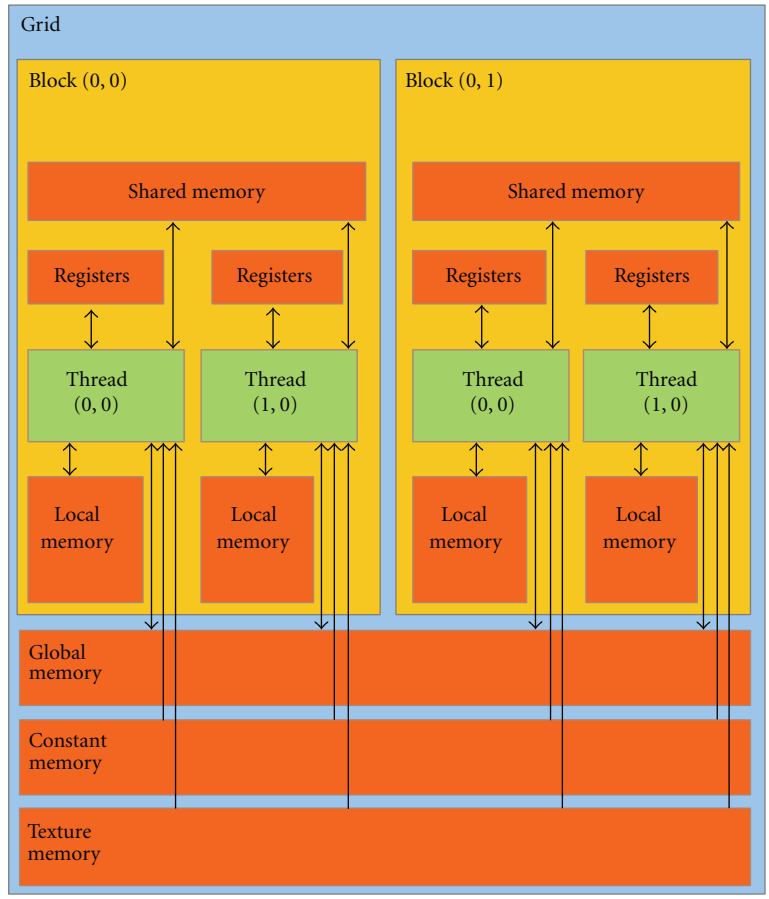


Figure 2: CUDA GPU memory model design [22].

Definition 1. A *type 2 fuzzy set*, denoted $\widetilde{A}$, is characterized by a type 2 membership function $\mu_{\widetilde{A}}(x,u)$ where $x \in X$ and $u \in J_x \subseteq [0,1]$, that is,

$$\widetilde{A} = \left\{ \left((x,u), \mu_{\widetilde{A}}(x,u) \right) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \right\} \quad (1)$$

or

$$\widetilde{A} = \int_{x \in X} \int_{u \in J_x} \frac{\mu_{\widetilde{A}}(x,u)}{(x,u)}, \quad J_x \subseteq [0,1] \quad (2)$$

in which $0 \le \mu_{\widetilde{A}}(x,u) \le 1$.

At each value of x, say $x = x'$, the 2D plane whose axes are u and $\mu_{\widetilde{A}}(x',u)$ is called a *vertical slice* of $\mu_{\widetilde{A}}(x,u)$. A *secondary membership function* is a vertical slice of $\mu_{\widetilde{A}}(x,u)$. It is $\mu_{\widetilde{A}}(x = x',u)$ for $x \in X$ and for all $u \in J_{x'} \subseteq [0,1]$, that is,

$$\mu_{\widetilde{A}}(x = x',u) \equiv \mu_{\widetilde{A}}(x') = \int_{u \in J_{x'}} \frac{f_{x'}(u)}{u}, \quad J_{x'} \subseteq [0,1] \quad (3)$$

in which $0 \le f_{x'}(u) \le 1$.

In manner of embedded fuzzy sets, a type 2 fuzzy sets [1] is union of its type 2 embedded set, that is,

$$\widetilde{A} = \sum_{j=1}^{n} \widetilde{A}_e^j, \quad (4)$$

where $n \equiv \prod_{i=1}^{N} M_i$ and $\widetilde{A}_e^j$ denoted the jth type 2 embedded set of $\widetilde{A}$, that is,

$$\widetilde{A}_e^j \equiv \left\{ \left(u_i^j, f_{x_i}\left(u_i^j\right) \right), \ i = 1,2,\ldots,N \right\}. \quad (5)$$

where $u_i^j \in \{u_{ik}, k = 1,\ldots,M_i\}$.

Type 2 fuzzy sets are called interval type 2 fuzzy sets if the secondary membership function $f_{x'}(u) = 1$, for all $u \in J_x$, that is, a type 2 fuzzy set is defined as follows.

Definition 2. An *interval type 2 fuzzy set* $\widetilde{A}$ is characterized by an interval type 2 membership function $\mu_{\widetilde{A}}(x,u) = 1$ where $x \in X$ and $u \in J_x \subseteq [0,1]$, that is,

$$\widetilde{A} = \{((x,u),1) \forall x \in X, \forall u \in J_x \subseteq [0,1]\}. \quad (6)$$

Uncertainty of $\widetilde{A}$, denoted FOU, is union of primary functions that is $\mathrm{FOU}(\widetilde{A}) = \bigcup_{x \in X} J_x$. Upper/lower bounds of membership function (UMF/LMF), denoted $\overline{\mu}_{\widetilde{A}}(x)$ and $\underline{\mu}_{\widetilde{A}}(x)$, of $\widetilde{A}$ are two type 1 membership function and bounds of FOU.

3.2. Interval Type 2 Fuzzy Logic Systems (IT2FLSs). The general type 2 fuzzy logic system is introduced as Figure 3. The output block of a type 2 fuzzy logic system consists of two blocks that are type-reduced and defuzzifier. The type-reduced block will map a type 2 fuzzy set to a type 1 fuzzy set, and the defuzzifier block will map a fuzzy to a crisp. The membership function of an interval type 2 fuzzy set is called FOU which is limited by two membership functions of a type 1 fuzzy set that are UMF and LMF (see Figure 4).

The combination of antecedents in a rule for IT2FLS is called firing strength process represented by the Figure 5.

In the IT2FLS, calculating process involves 5 steps to getting outputs: fuzzification, combining the antecedents (apply fuzzy operators or implication function), aggregation, and defuzzification.

Because each pattern has a membership interval as the upper $\overline{\mu}_{\widetilde{A}}(x)$ and the lower $\underline{\mu}_{\widetilde{A}}(x)$, each centroid of a cluster is represented by the interval between c_L and c_R. Now, we will represent an iterative algorithm to find c_L and c_R as follows.

Step 1. Calculate θ_i by the following equation:

$$\theta_i = \frac{1}{2}\left[\overline{\mu}(x_i) + \underline{\mu}(x_i)\right]. \quad (7)$$

Step 2. Calculate c' as follows:

$$c' = c(\theta_1, \theta_2, \ldots, \theta_N) = \frac{\sum_{i=1}^{N} x_i * \theta_i}{\sum_{i=1}^{N} \theta_i}. \quad (8)$$

Step 3. Find k such that $x_k \le c' \le x_{k+1}$.

Step 4. Calculate c'' by following equation: in case c'' is used for finding c_L

$$c'' = \frac{\sum_{i=1}^{k} x_i \overline{\mu}(x_i) + \sum_{i=k+1}^{N} x_i \underline{\mu}(x_i)}{\sum_{i=1}^{k} \overline{\mu}(x_i) + \sum_{i=k+1}^{N} \underline{\mu}(x_i)}. \quad (9)$$

In case c'' is used for finding c_R, then

$$c'' = \frac{\sum_{i=1}^{k} x_i \underline{\mu}(x_i) + \sum_{i=k+1}^{N} x_i \overline{\mu}(x_i)}{\sum_{i=1}^{k} \underline{\mu_A}(x_i) + \sum_{i=k+1}^{N} \overline{\mu}(x_i)}. \quad (10)$$

Step 5. If $c' = c''$ go to Step 6 else set $c' = c''$, then back to Step 3.

Step 6. Set $c_L = c'$ or $c_R = c'$.

Finally, compute the mean of centroid, y, as

$$y = \frac{c_R + c_L}{2}. \quad (11)$$

4. Speedup of IT2FLS Using GPU and CUDA

The first step in IT2FLS on the GPU is selection of memory types and sizes. This is a critical step, the choice of format and type dictate performance. Memory should be allocated such that sequential access (of read and write operations) is as possible as the algorithm will permit.

Let the number of inputs be N, the number of parameters that define a membership function be P, the number of rules be R and the discretization rate be S. Inputs are stored on the GPU as a one-dimensional array of size N (see Figure 6).

The consequences are a CPU two-dimensional array of size $R \times P$. They are used only on the CPU when calculating the discrete fuzzy set membership values.

The antecedents are a two-dimensional array on the GPU of size $R \times (N \times P)$.

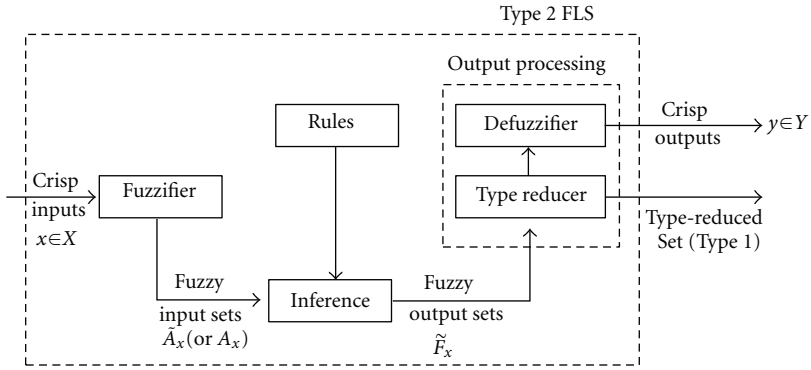


Figure 3: Diagram of type 2 fuzzy logic system [4].

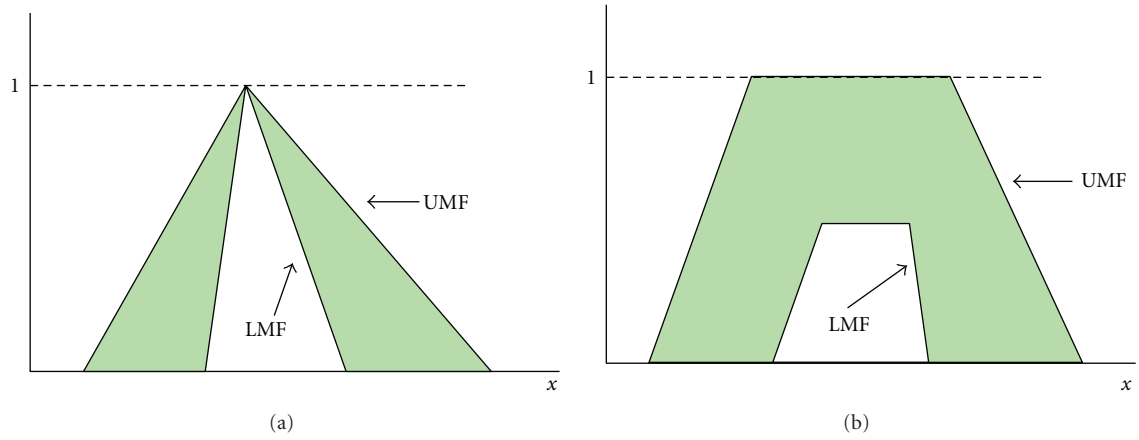


Figure 4: The membership function of an interval type 2 fuzzy set [1].

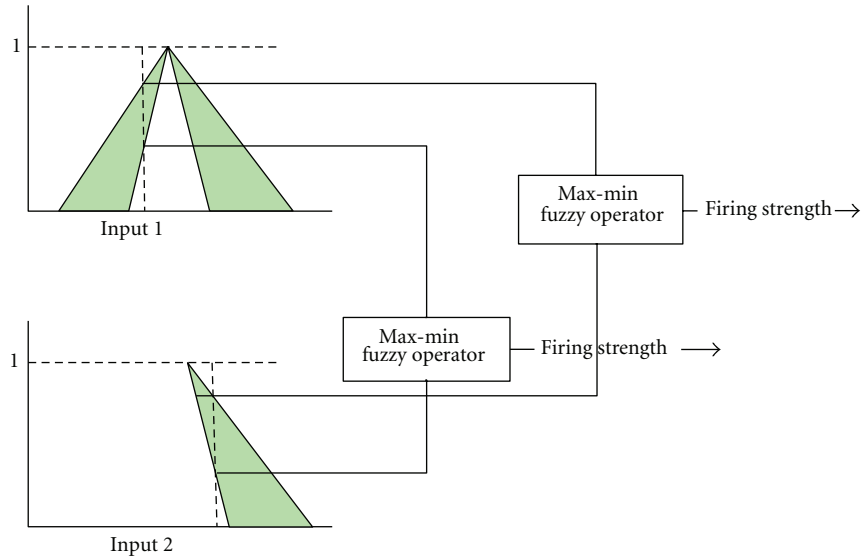


Figure 5: The combination of antecedents in a rule for IT2FLS [23].

	N			
1	X_1	X_2	$\cdots$	X_N

Figure 6: Input vector.

The fired antecedents are an R one-dimensional array on the GPU, which stores the result of combining the antecedents of each rule (see Figures 7, 8, and 9). The last memory layout is the discretized consequent, which is an $S \times R$ matrix created on the GPU.

The inputs and antecedents are of type texture memory because they do not change during the firing of a FLS, but could change between consecutive firings of a FLS and need to be updated. We proposed the GPU program flow diagram for a CUDA application computing a IT2FLS in Figure 10.

In IT2FLS, we have to calculate two values for two membership functions that are UMF and LMF. The first step is a kernel that fuzzifies the inputs and combines the antecedents. The next steps are implication and a process which is responsible for aggregating the rule outputs. The last GPU kernel is the defuzzification step.

The first kernel reads from the inputs and antecedents textures and stores its results in the fired antecedent's global memory section. All inputs are sampled for each rule, the rth rule samples the rth row in the antecedent's memory, membership values are calculated, and the minimum of the antecedents is computed and stored in the rth row of the fired antecedent's memory region. There are B blocks used by this kernel, partially because there is a limit in terms of the number of threads that can be created per block (current max is 512 threads). Also, one must consider the number of threads and the required amount of register and local memory needed by a kernel to avoid memory overflow. This information can be found per each GPU. We limited the number of threads per block to 128 (an empirical value found by trying different block and thread profiles for a system that has two inputs and trapezoidal membership functions). The general goal of a kernel should be to fetch a small number of data points, and it should have high arithmetic intensity. This is the reason why only a few memory fetches per thread are made, and the membership calculations and combination step is performed in a single kernel.

The next steps are implication and rule aggregation kernels. At first, one might imagine that using two kernels to calculate the implication results and rule aggregation would be desirable. However, the implication kernel, which simply calculates the minimum between the respective combined antecedent results and the discretized consequent, is inefficient. As stated above, the ratio of arithmetic operations to memory operations is important. We want more arithmetic intensity than memory access in a kernel. Attempt to minimize the number of global memory samples, we perform implication in the first step of reduction. Reduction, in this context, is the repeated application of an operation to a series of elements to produce a single scalar result. In the case of rule aggregation, this is the application of the maximum

	P_1			
	C_{11}	C_{12}	$\cdots$	C_{1P1}
	C_{21}	C_{22}	$\cdots$	C_{2P1}
R	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	C_{R1}	C_{R2}	$\cdots$	C_{RP1}

Figure 7: Consequent.

	$M = N \times P$			
	A_{11}	A_{12}	$\cdots$	A_{1M}
	A_{21}	A_{22}	$\cdots$	A_{2M}
R	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	A_{R1}	A_{R2}	$\cdots$	A_{RM}

Figure 8: Antecedent.

	S = sample rate			
	D_{11}	D_{12}	$\cdots$	D_{1S}
	D_{21}	D_{22}	$\cdots$	D_{2S}
R	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	D_{R1}	D_{R2}	$\cdots$	D_{RS}

Figure 9: Discretized consequent.

operator over each discrete consequent sample point for each rule. The advantage of GPU reduction is that it takes advantage of the parallel processing units to perform a divide and conquer strategy. And the last step is defuzzification kernel. As described above, rule output aggregation and defuzzification reduction are used for IT2FLS on the GPU.

The output of rule output aggregation is two rows in the discretized consequent global memory array. The defuzzifier step is done by the Karnik-Mendel algorithms with two inputs that are rule_combine_UMF and rule_combine_LMF with two outputs y_l and y_r, respectively. The crisp output y is calculated by the formula $y = (y_l + y_r)/2$.

The steps for finding y_l and y_r on GPU (Notation: Rule_Combine_UMF (i) = $\overline{\mu_A}(x_i)$ and Rule_Combine_LMF (i) = $\underline{\mu_A}(x_i)$, N = sample rate) as follows

Step 1. Calculate θ_i on GPU by the following equation:

$$\theta_i = \frac{1}{2}\left[\overline{\mu_A}(x_i) + \underline{\mu_A}(x_i)\right]. \tag{12}$$

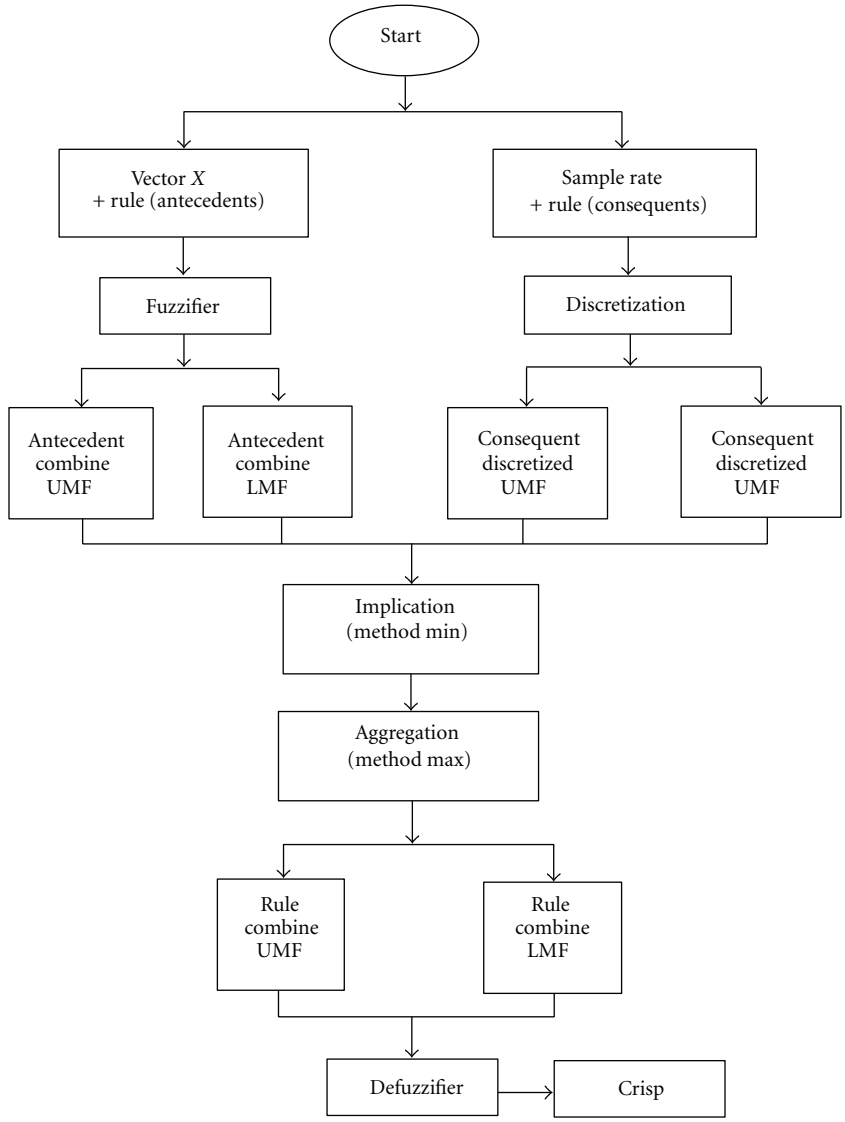


Figure 10: IT2FLS diagram for a CUDA application on the GPU.

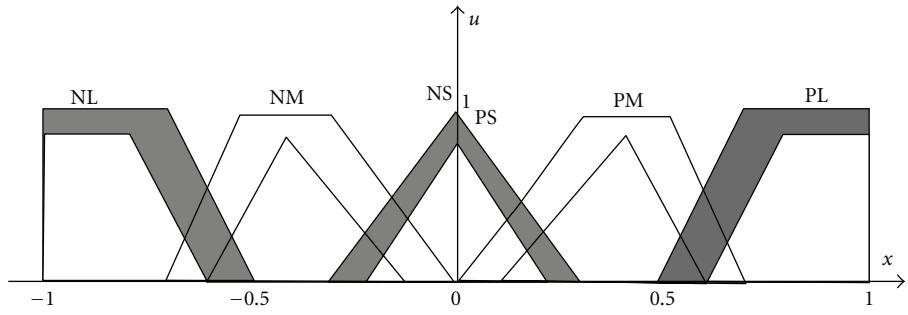


Figure 11: Membership grades of FDR.

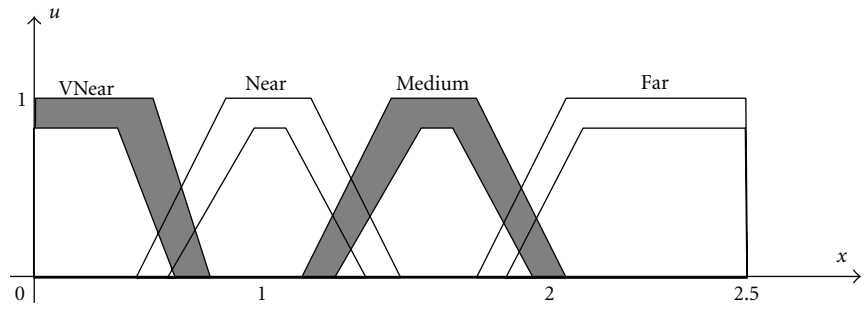


Figure 12: Membership grades of Range.

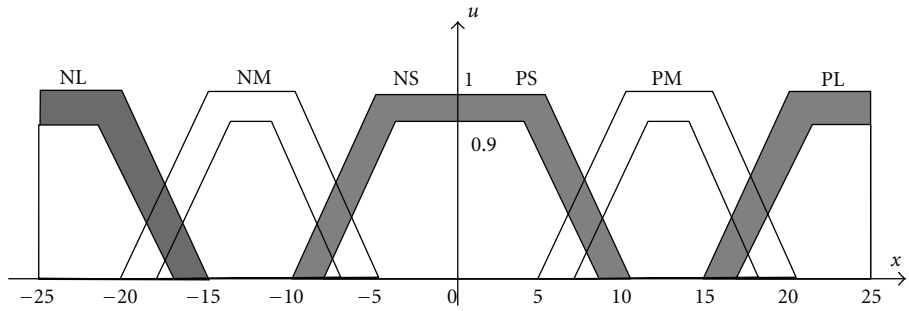


Figure 13: Membership grades of AoD.

Table 1: The rule base of collision avoidance behavior.

FDR	Range	AoD	FDR	Range	AoD
NS	VN	PL	PS	VN	NL
NS	N	PL	PS	N	NL
NS	M	PM	PS	M	NM
NS	F	PS	PS	F	NS
NM	VN	PM	PM	VN	NM
NM	N	PM	PM	N	NM
NM	M	PM	PM	M	NM
NM	F	PS	PM	F	NS
NL	VN	PM	PL	VN	NM
NL	N	PM	PL	N	NM
NL	M	PS	PL	M	NS
NL	F	PS	PL	F	NS

Step 2. Calculate c' on GPU as follows:

$$c' = c(\theta_1, \theta_2, \ldots, \theta_N) = \frac{\sum_{i=1}^{N} x_i * \theta_i}{\sum_{i=1}^{N} \theta_i}. \tag{13}$$

Next, copy c' to host memory.

Step 3. Find k such that $x_k \le c' \le x_{k+1}$ (calculated on CPU).

Step 4. Calculate c'' on GPU by following equation. In case c'' is used for finding y_l

$$c'' = \frac{\sum_{i=1}^{k} x_i \overline{\mu_A}(x_i) + \sum_{i=k+1}^{N} x_i \underline{\mu_A}(x_i)}{\sum_{i=1}^{k} \overline{\mu_A}(x_i) + \sum_{i=k+1}^{N} \underline{\mu_A}(x_i)}. \tag{14}$$

In case c'' is used for finding y_r, consider

$$c'' = \frac{\sum_{i=1}^{k} x_i \underline{\mu_A}(x_i) + \sum_{i=k+1}^{N} x_i \overline{\mu_A}(x_i)}{\sum_{i=1}^{k} \underline{\mu_A}(x_i) + \sum_{i=k+1}^{N} \overline{\mu_A}(x_i)}. \tag{15}$$

Next, copy c'' to host memory.

Step 5. If $c' = c''$ go to Step 6 else set $c' = c''$, then back to Step 3 (calculated on CPU).

Step 6. Set $y_l = c'$ or $y_r = c'$.

5. Experiments

5.1. Problems. We implement IT2FLS with collision avoidance behavior of robot navigation. The fuzzy logic systems have two inputs: the extended fuzzy directional relation (FDR) [24] and range to obstacle; the output is angle of deviation (AoD). The fuzzy rule has the form as follows.

IF FDR is $\widetilde{A}_i$ AND Range is $\widetilde{B}_i$ THEN AoD is $\widetilde{C}_i$, where $\widetilde{A}_i$, $\widetilde{B}_i$, and $\widetilde{C}_i$ are type 2 fuzzy sets of antecedent and consequent, respectively.

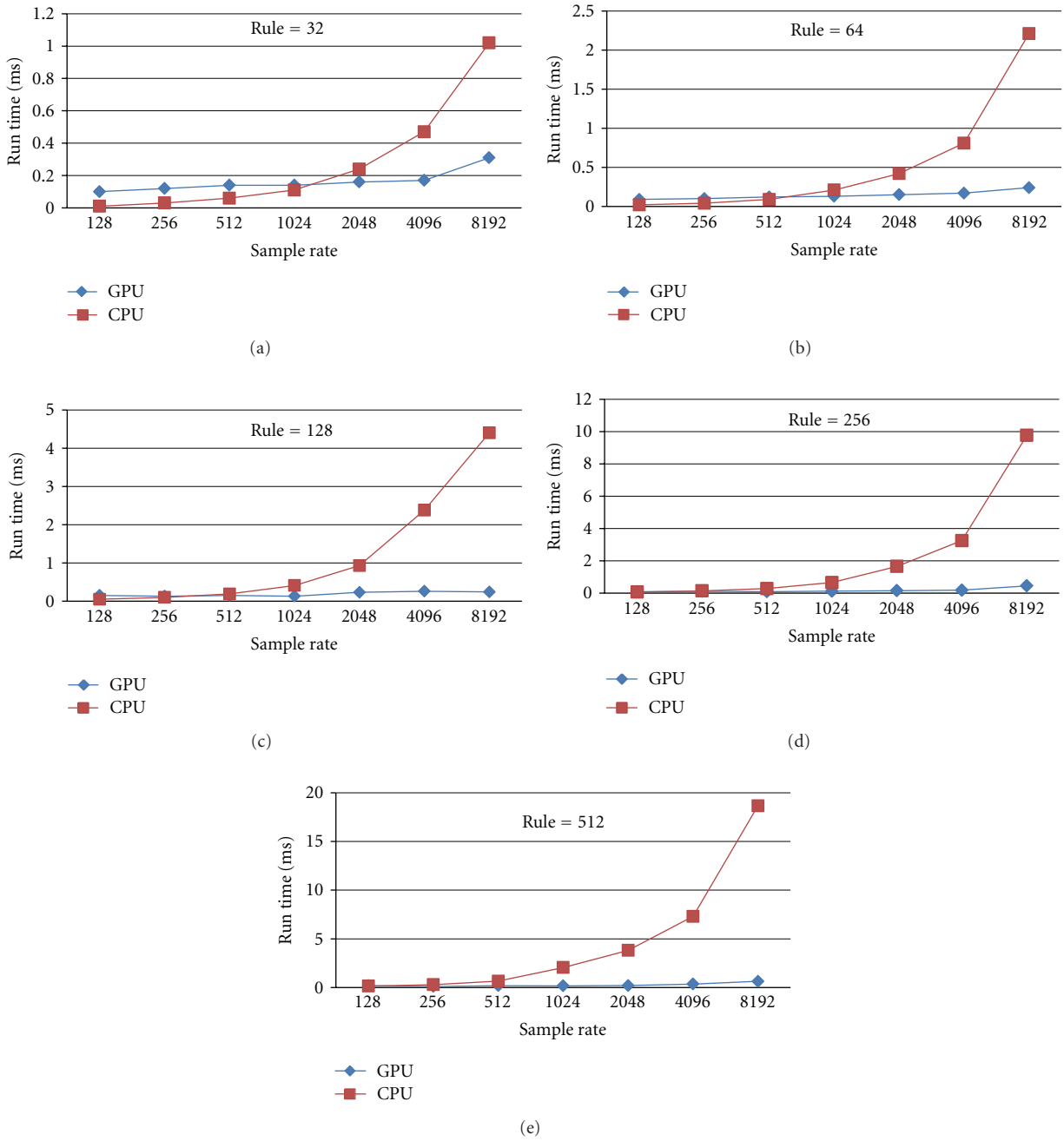


Figure 14: Run-time graph of the problem implementation.

The fuzzy directional relation has six linguistic values (NLarge, NMedium, NSmall, PSmall, PMedium, and PLarge). The range from robot to obstacle is divided into four subsets: VNear, Near, Medium, and Far. The output of fuzzy if-then is a linguistic variable representing for angle of deviation and has six linguistic variables the same the fuzzy directional relation with the different membership functions. Linguistic values are interval type 2 fuzzy subsets that membership functions are described in Figures 11, 12, and 13. The problem is built with 24 rules given by the following Table 1.

5.2. Experiments. The performance of the GPU implementation of a IT2FLS was compared to a CPU implementation. The problem is written in C/C++ console format and be installed on the Microsoft Visual Studio 2008, and it was performed on computers with the operating system windows 7 32 bit and nVIDIA CUDA support with specifications.

CPU was the Core i3-2310 M 2.1 GHz, the system had 2 GB of system RAM (DDR3).

GPU was an nVIDIA Gerforce GT 540 M graphics card with 96 CUDA Core, 1 GB of texture memory, and PCI Express X16.

Table 2: CPU/GPU performance ratio.

R	S						
	128	256	512	1024	2048	4096	8192
32	0.1	0.24	0.41	0.76	1.47	2.64	3.22
64	0.21	0.39	0.73	1.58	2.7	4.69	8.99
128	0.32	0.76	1.25	3.07	3.95	9.01	18.26
256	0.72	1.19	2.95	5.35	10.7	17.56	21.3
512	0.77	2.43	3.32	12.57	18.43	20.51	29.3

The number of inputs was fixed to 2, the number of rules was varied between 32, 64, 128, 256, and 512, and sample rate was varied between 256, 512, 1024, 2048, 4096, and 8192.

We take the ratio of CPU versus GPU performance. A value below 1 indicated that the CPU is performing best, and value above 1 indicates the GPU is performing best. The CPU/GPU performance ratios for the IT2FLS are given in Table 2 and run-time graph of the problem implementation was shown in Figure 14.

6. Conclusion

As demonstrated in this paper, the implementation of interval type 2 FLS on a GPU without the use of a graphics API which can be used by any researcher with knowledge of C/C++. We have demonstrated that the CPU outperforms the GPU for small systems. As the number of rules and sample rate grow, the GPU outperforms the CPU. There is a switch point in the performance ratio matrices (Table 2) that indicates when the GPU is more efficient than the CPU. In the case that sample rate is 8192 and rule is 512, the GPU runs approximately 30 times faster on the computer.

Future work will look at to extend interval type 2 FLS to the generalised type 2 FLS and applying to various applications.

Acknowledgment

This paper is sponsored by Intelligent Robot Project at LQDTU and the Research Fund RFit@LQDTU, Faculty of Information Technology, Le Quy Don University.

References

[1] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999.

[2] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, no. 1–4, pp. 195–220, 2001.

[3] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, 2000.

[4] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, pp. 808–821, 2006.

[5] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 178, no. 9, pp. 2224–2236, 2008.

[6] L. T. Ngo, L. T. Pham, P. H. Nguyen, and K. Hirota, "On approximate representation of type-2 fuzzy sets using triangulated irregular network," in *Foundations of Fuzzy Logic and Soft Computing*, vol. 4529 of *Lecture Notes in Computer Science*, pp. 584–593, Springer, 2007.

[7] L. T. Ngo, L. T. Pham, P. H. Nguyen, and K. Hirota, "Refinement geometric algorithms for type-2 fuzzy set operations," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 866–871, August 2009.

[8] L. T. Ngo, "Refinement CTIN for general type-2 Fuzzy logic systems," in *Proceedings of the IEEE International Conference on Fuzzy Systems (IEEE-FUZZ '11)*, pp. 1225–1232, Hanoi, Vietnam, 2011.

[9] J. T. Starczewski, "Efficient triangular type-2 fuzzy logic systems," *International Journal of Approximate Reasoning*, vol. 50, no. 5, pp. 799–811, 2009.

[10] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 524–539, 2004.

[11] O. Castillo and P. Melin, "A review on the design and optimization of interval type-2 fuzzy controllers," *Applied Soft Computing*, vol. 12, no. 4, pp. 1267–1278, 2012.

[12] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Modelling and simulation of the defuzzification stage of a type-2 fuzzy controller using VHDL code," *Control and Intelligent Systems*, vol. 39, no. 1, pp. 33–40, 2011.

[13] Y. Maldonado, O. Castillo, and P. Melin, "Optimization of membership functions for an incremental fuzzy PD control based on genetic algorithms," *Soft Computing for Intelligent Control and Mobile Robotics*, vol. 318, pp. 195–211, 2011.

[14] R. Sepúlveda, O. Montiel-Ross, O. Castillo, and P. Melin, "Embedding a KM type reducer for high speed fuzzy controller into an FPGA," *Applied Soft Computing*, vol. 12, no. 3, pp. 988–998, 2012.

[15] D. T. Anderson, R. H. Luke, and J. M. Keller, "Speedup of fuzzy clustering through stream processing on graphics processing units," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 4, pp. 1101–1106, 2008.

[16] D. Anderson, R. H. Luke, and J. M. Keller, "Incorporation of non-euclidean distance metrics into fuzzy clustering on graphics processing units," in *Proceedings of the Inertial Fusion Sciences and Applications (IFSA '07)*, vol. 41, pp. 128–139, 2007.

[17] K. Sejun and C. Donald, "A GPU based parallel hierarchical fuzzy ART clustering," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 2778–2782, Rolla, Mo, USA, 2011.

[18] I. Chiosa and A. Kolb, "GPU-based multilevel clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 132–145, 2011.

[19] F. Chia, C. Teng, and Y. Wei, "Speedup of implementing fuzzy neural networks with high-dimensional inputs through parallel processing on graphic processing units," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 4, pp. 717–728.

[20] N. Harvey, R. Luke, J. M. Keller, and D. Anderson, "Speedup of fuzzy logic through stream processing on graphics processing units," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 3809–3815, June 2008.

[21] D. Anderson, "Parallelisation of fuzzy inference on a graphics processor unit using the compute unified device architecture," in *Proceedings of the UK Workshop on Computational Intelligence (UKCI '08)*, pp. 1–6, 2008.

[22] M. Harris, 18 March 2008, Optimizing Parallel Reduction in CUDA, NVIDIA Whitepaper, http://www.nvidia.com/object/cuda/sample/dat/a%20-%20parallel.html.

[23] R. Imam and B. Kharisma, "Design of interval type-2 fuzzy logic based power system stabilizer," *International Journal of Electrical and Electronics Engineering*, vol. 3, no. 10, pp. 593–600, 2009.

[24] L. T. Ngo, L. T. Pham, and P. H. Nguyen, "Extending fuzzy directional relationship and applying for mobile robot collision avoidance behavior," *International Journal of Advanced Computational Intelligence & Intelligent Informatics*, vol. 10, no. 4, pp. 444–450, 2006.

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 219602, 10 pages
doi:10.1155/2012/219602

Research Article

WLAN Cell Handoff Latency Abatement Using an FPGA Fuzzy Logic Algorithm Implementation

Roberto Sepúlveda, Oscar Montiel-Ross, Jorge Quiñones-Rivera, and Ernesto E. Quiroz

Instituto Politécnico Nacional-CITEDI, Avenida del Parque No. 1310, 22510 Tijuana, BC, Mexico

Correspondence should be addressed to Roberto Sepúlveda, r.sepulveda@ieee.org

Received 5 May 2012; Accepted 29 May 2012

Academic Editor: Oscar Castillo

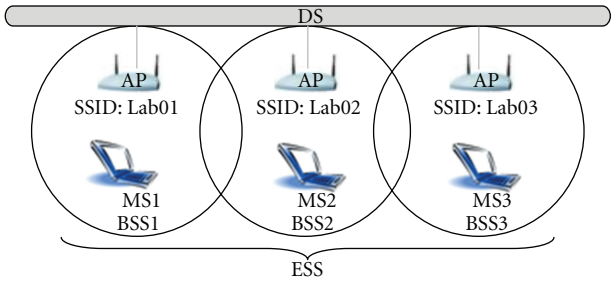
Following the path toward 4 G set by its wireless siblings LTE and WiMax, IEEE 802.11 technology, universally known as WiFi, is evolving to become a high data rate QoS-enabled mobile platform. The IEEE 802.11n standard yields data rates up to 450 Mbp s and the 802.11e standard ensures proficient QoS for real-time applications. Still in need of better performance, multicell environments that provide extended coverage allow the mobile station nomadic passage beyond a single cell by means of cell dissociation-association process known as handoff. This process poses a challenge for real-time applications like voice over IP (150 ms maximum delay) and video (200–400 ms) sessions, to give the user a seamless cell-crossing without data loss or session breakage. It presented an approach of a predictive fuzzy Logic controller to reduce the channel scanning process to a tenth of the standard time, and its efficient FPGA implementation to speed up the processing time. The algorithm of the fuzzy controller was implemented in C language. Experimental results are provided.

1. Introduction

WiFi (Wireless Fidelity) wireless connectivity keeps permeating virtually in all categories of consumer electronics devices. Besides the traditional application in laptops, tablets, and dual-mode (cellular-WiFi) handsets, WiFi connectivity is lurking into TVs, media players, gaming consoles, connected home, and so forth. Production of WiFi devices reached nearly 1.1 billion in 2011 and is expected to double by 2015. The growth is a solid 25–39% in health, fitness, medical applications, smart meters, and automation products; a staggering 109% increase was predicted in automotive applications such as infotainment systems, navigation, and traffic monitoring [1].

WiFi relies on the IEEE 802.11 [2] radio standards, the WiFi Protected Access (WPA) and WPA2 security standards, and the EAP (Extensible Authentication Protocol) authentication standard. A WiFi Access Point (AP) provides wireless connectivity to Mobile Stations (MS) within its cell coverage (30–110 yards), varying by commercial implementation. A multi-cell ensemble can be fitted to a particular setting to cover a wider area with multiple APs (Figure 1). MS handoff (HO) from one cell to another provides seamless connectivity to itinerant users moving beyond the radio range of its currently associated AP to enter a neighboring AP's basic service set (coverage area). During the handoff process, management frames are exchanged between the MS and the AP. Also the APs involved exchange context information pertaining to the MS. Table 1 presents the standards that represent the generations in the evolution of WiFi [3].

Voice was not originally supported by WiFi, but the convergence trend brought about by the IP protocol thrusts voice over IP (VoIP) into the digital multimedia stream. Also, the great advancements in VoIP compression technology and the deployment of carrier-class VoIP networks have made possible extremely low prices. Adhering to the VoIP drive, IEEE 802.11n incorporates quality of service (QoS) mechanisms on a per cell basis, with stringent performance requirements, that is: (a) packet loss of less than 1% with no burst losses; (b) latency of less than 50 ms; (c) maximum jitter of less than 50 ms; (d) VoIP precedence over any other data stream being handled by the AP. Nevertheless, HO is not among the high priority issues. An inefficiently handoff process might cause call interruption or at best, uncomfortable loss of information. Situations that can push



DS: Distribution system
AP: Access point
SSID: Service set identifier
MS: Mobile station
BSS: Basic service set
ESS: Extended service set

Figure 1: Extended and basic service sets coverage.

Table 1: Wi-Fi generations.

Wi-Fi technology	Frequency band	Maximum data rate
802.11a	5 GHz	54 Mbps
802.11b	2.4 GHz	11 Mbps
802.11g	2.4 GHz	54 Mbps
802.11n	2.4 GHz, 5 GHz, 2.4 or 5 GHz (selectable), or 2.4 and 5 GHz (concurrent)	450 Mbps

the HO process to its limits are: (a) WLAN cells have a small coverage area, and a fast moving MS may produce frequent, short-time interval handoffs. (b) There is a latency (≈300 msec.) [4] involved in the handoff process during which the MS is unable to send or receive any kind of traffic. (c) VoIP requires one-way end-to-end delay of less than 150 ms [5].

Faster processing time during HO is desirable. The continuous breakthroughs in Very Large Scale Integrated Circuits (VLSI), mainly in Field-Programmable Gate Array (FPGA) and the Applied Specific Integrated Circuit (ASIC) domains, and also in the development of new programming tools that allow to undertake complex digital designs to creation in a very small time affords to design and implement high-performance systems embedded into an integrated circuit. Other features about FPGA are that they consume low power and can be reprogrammable in field; hence, there is an increasing interest in using FPGA devices to design digital controllers, and a growing interest in control systems that require a real time operation based on fuzzy logic. Nowadays, the studies and proposals to implement fuzzy systems into an FPGA keep adding [6–8], including interesting FPGA implementations of type 1 fuzzy inference systems (FIS) in electrical vehicles, such as [9]. Some proposals implement type 2 FIS [10–12], as well as others that focus on the software development for coding a high-speed defuzzification stage for a type 2 FIS [13]. More recent works include [14], where an interval type 2 FIS Karnik-Mendel-type reducer is designed, tested, and implemented.

In order for QoS delay constraints for VoIP to be fulfilled during an intercell handover, we implement a fuzzy logic fast handoff algorithm in an FPGA, which instead of scanning multiple channels predicts the best fitted channel and scans only the selected channel, reducing the lengthier part of the HO process up to an eleventh, since eleven is the maximum number of adjacent cell channels the standard allows. The following sections of the paper are organized as follows. Section 2 explains 802.11 handoff procedure, a selection of other variants looking to reduce the time consumed in the process, and our solution. Section 3 discusses the FPGA implementation. Section 4 presents the simulation results and Section 5 concludes the paper.

2. Intercell Handoff Procedures

2.1. 802.11 Handoff Standard. The main requirements an HO procedure must fulfill are low latency, scalability, minimum drop-off and fast recovery, QoS (maintained or renegotiated), and security.

When an MS approaches the limits of its BSS, the intensity and quality of its signal deteriorate, approaching a link breaking point. The AP's radio signals are severely affected by various environmental factors, such as distance-related attenuation, obstacle-induced fading, multipath signal aggregation, and other radiofrequency sources interference. As the MS's Received Signal Strength (RSS) from the source AP diminishes, the MS receives strong signals from neighboring APs, so that when the local signal faints to a predefined threshold (i.e., 80 dBm), the HO process is initiated [15]; see Figure 2.

IEEE 802.11 contemplates two channel configurations to carry on the handoff procedure: single-channel roaming and multichannel roaming. In this paper we will be dealing with the latter, since the channel-scanning process takes longer than the former.

2.1.1. Multichannel Roaming. Each AP is assigned a beacon channel, to transmit a reference signal at 100 ms intervals [16]. Up to eleven channels can be used in an Extended Service Set (ESS) simultaneously (limit stated in 802.11 standard) [17]. A polling scheme is used by the MS to scan each beacon-frame channel carrier generated by all surrounding APs. When the signal quality falls below the "start cell search" threshold, the MS sends a probe by each channel, requesting an immediate transmission of a beacon signal (Figure 3). Neighboring APs send a response frame in a sequential fashion, so the MS can perform a real time measure of the received power from all APs. When an AP's signal is better than the threshold, the MS initiates the transference to the selected AP's neighboring cell. The target AP authenticates, associates with the MS, and allocates resources. Following communicates with the source AP via an inter-accesspoint protocol [17]. Finally, the source AP disassociates from the MS. The time elapsed in the HO execution is variable, depending on the number of channels (cell ensemble) present for scanning. When the eleven channel are used, the time consumed is around 300 ms [4].

2.2. Related Work. Mishra et al. [18] analyze handoff latency at the link layer, demonstrating that multichannel-scanning

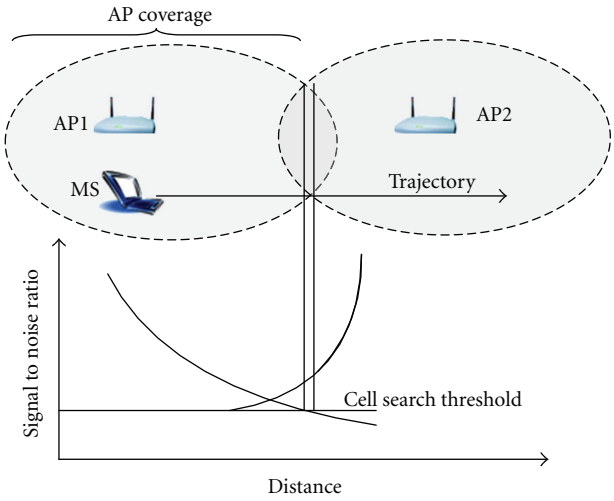


Figure 2: Intercell overlap zone for handoff realization.

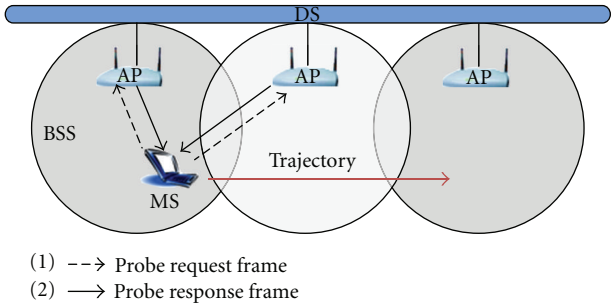


Figure 3: Scan sweep.

contributes with most of the total handoff delay, yielding poor QoS for time-sensitive applications. Shin et al. [19] use a scanning algorithm to select a subset of channels and build a channel mask. This mask is used in the next handoff to scan only the predefined subset reducing the scanning delay by 30% to 60% compared to the standard procedure. Li et al. [20] use a neighbors' graphics caching mechanism. The mobile station will know beforehand the channels used by the neighboring access points, so there is no need to explore all available channels, reducing scanning latency. Chang et al. [21] assess the average change in the MS's RSS, pinpointing the more appropriate AP to serve the handoff. Song et al. [22] introduce improvements to the mobile IPv6 fast handover protocol in a subnet to subnet MS transference. Purushothaman and Roy's work [4] reduces the number of channels to scan by using a client-based database which stores information about the APs' channel numbers with higher RSS. Ong and Khan [23] eliminate the channel sweep stage based on the traditional power signal metrics. The HO triggering reference is the number of VoIP packets lost, which should not exceed 2% of the total sent in a period of time. This approach cuts 90% of the 802.11 standard handoff time [4]. Nevertheless, the downside is the cost of greater network complexity brought about by the need of an increased network-wide information exchange.

Some works have relied on the use of fuzzy logic (FL) in diverse aspects of the WLAN operation. Patil and Kolte [24] describe a five-parameter FL algorithm for handoff optimization. Nevertheless, it is generic and does not apply to the 802.11 WLAN case. Gharehbaghi and Badamchizadeh [25] perform a comparison of four FL algorithms for congestion control, a special high volume packet arrival situation of the target AP which causes a service denial to the approaching MS.

Our proposal relies on the link layer received-signal-power detection already in use, and the calculation of the direction of the MS in relation to the neighboring APs (used in [20]), to build a two-input Predictive Fuzzy Logic Control (PFLC) designed to compute in advance in which AP has the highest aptitude value to admit the MS when handoff initiation becomes necessary. So, in order to conform to the IEEE.802.11 triggering process, only one channel is scanned. Our proposal, thus, approaches the time reduction obtained by [23], but modifications are confined to the MS, instead of the whole network.

3. Cell Handoff Fuzzy Logic Controller Design and Implementation

3.1. Predictive Fuzzy Logic Controller. Figure 4 illustrates the Predictive Fuzzy Logic Controller (PFLC) participation in the HO process. At 100 msec., time intervals the APs broadcast beacon signals. The MS receives the RSS information from each channel and the PFLC performs an analysis that yields the channel number identifying the AP with the highest aptitude value at that time. In this manner, the MS has an updated knowledge of the best fitted AP to associate with. So, when the AP1 signal quality falls to the search threshold, triggering the handoff, the MS knows, among the eleven contiguous AP's channels, that AP2 has the higher aptitude. The standard's scanning operation is performed only once on the preselected channel.

A Mamdani-type fuzzy system was designed to predict, from a group of cells, which AP is best fitted to serve the itinerant MS. Figure 4 illustrates the two-input-one-output fuzzy controller; the same fuzzy system can be seen with more details in Figure 11. The first entry represents the Average Signal Intensity (ASI), calculated at two-second intervals from the beacon signal received by the mobile station every 100 msec. ASI constitutes a major metrics typically used in wireless systems to measure signal quality for purposes of performing the handoff. While the MS beacon signal does not fall to cell search threshold (set to 80 dBm for this work), the handoff process is not triggered. The second input is the Signal Intensity Variation (SIV) parameter that provides information on the direction of the MS with respect to the AP (approaching or distancing).

Both inputs feed the fuzzy controller, which based on historic data and knowledge rules allow it to estimate the aptitude value of that particular AP. The same calculation is performed for al AP channels, and the highest aptitude output indicates which will be the target AP.

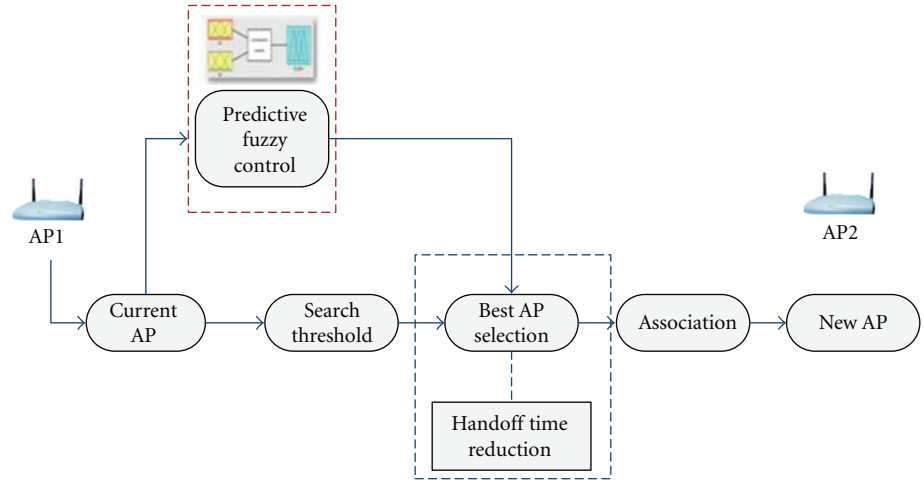


Figure 4: Predictive handoff scheme.

The inference mechanism applied is "max–min," widely known as the Mamdani method, characterized by its implementation simplicity and effectiveness. Also, the centroid method is used, because it provides more representative results from the output-weighted values of various membership functions.

The methodology to define membership functions for the three variables ranges (two inputs, one output) is described as follows.

(i) Input Variable: Average Signal Intensity. Four membership functions were set forth for ASI: low, medium, good, and excellent. The first and last are trapezoidal and the rest triangular, as shown in Figure 5.

The choice of the linguistic terms correspond to the quality of received signal.

The expression proposed by Chang et al. [21] is used to obtain the ASI received by the mobile station.

$$\mathrm{ASI}(t) = \frac{\sum_{t=1}^{n} \mathrm{SSbeacon}(t,i)}{n}. \quad (1)$$

The universe of discourse proposed for the ASI variable is shown in Figure 5, ASI is expressed by (1), where t is the current time, n represents the number of beacon frames received at time t, and finally SSbeacon (t, i) represents the intensity of the signal received at time "$t - i$" (Figure 6).

(ii) Input Variable: Signal Intensity Variation (SIV). For the relative position of an MS with respect to a specific AP, SIV represents the rate of change of two ASI readings made two seconds apart. Three membership functions were defined for SIV, whose linguistic terms are "negative," "zero", and "positive," being two of trapezoidal shape, and one triangular as illustrated in Figure 7.

The expression to calculate SIV is given in (2). SSa(t) is the ASI value at the present time, and SSa $(t-k)$ represents an

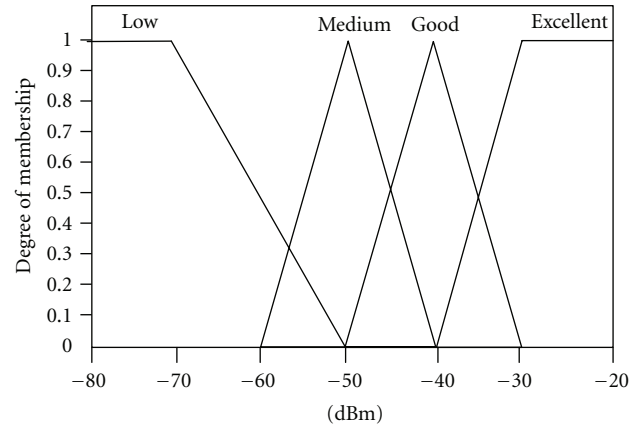


Figure 5: Average signal intensity in dBm.

ASI reading $k = 2$ seconds before. The range for this variable is set to $(-3, 3)$

$$\mathrm{SIV}(t) = \frac{\mathrm{SSa}(t) - \mathrm{SSa}(t-k)}{t - (t-k)}. \quad (2)$$

Thus, the variation of signal intensity contains information about the speed of movement of the MS and about direction of the MS in reference to the AP (Figure 8). A positive value is indicative of displacement towards the AP and a negative value, of moving apart.

The traditional handoff scheme considers only the ASI, whereas ASI and SIV yield a better picture of the MS behavior.

(iii) Variable Output: Aptitude. Aptitude represents the decision taken on the basis of historic data of the two inputs, ASI and SIV. Five membership functions were considered appropriate for this variable: negative, small negative, zero, small positive, and positive, as shown in Figure 9. The range is $(-2, 2)$.

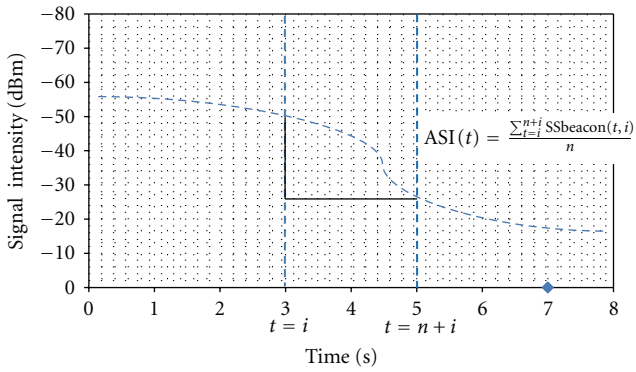


Figure 6: Interval time and the formula for the ASI calculation.

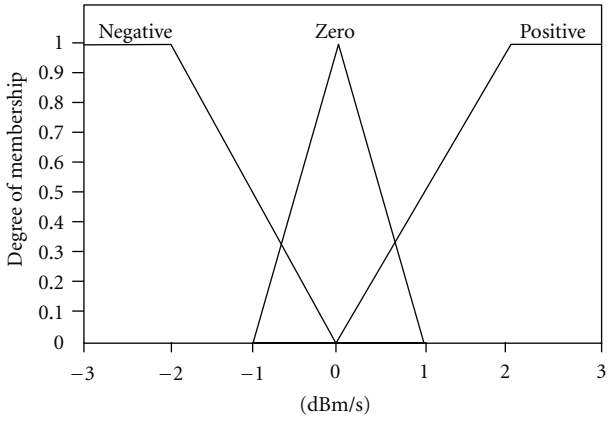


Figure 7: Signal intensity variation.

The fuzzy system's decision-making capability is defined by its knowledge base, comprised by 12 rules.

(1) If ASI is excellent and SIV is positive then aptitude is positive.

(2) If ASI is excellent and SIV is zero then aptitude is positive.

(3) If ASI is excellent and SIV is negative then aptitude is small positive.

(4) If ASI is good and SIV is positive then aptitude is positive.

(5) If ASI is good and SIV is zero then aptitude is small positive.

(6) If ASI is good and SIV is negative then aptitude is zero.

(7) If ASI is media and SIV is positive then aptitude is small positive.

(8) If ASI is medium and SIV is zero then aptitude is zero.

(9) If ASI is medium and SIV is negative then aptitude is small negative.

(10) If ASI is low and SIV is positive then aptitude is zero.

(11) If ASI is d Low own and SIV is zero then aptitude is small negative.

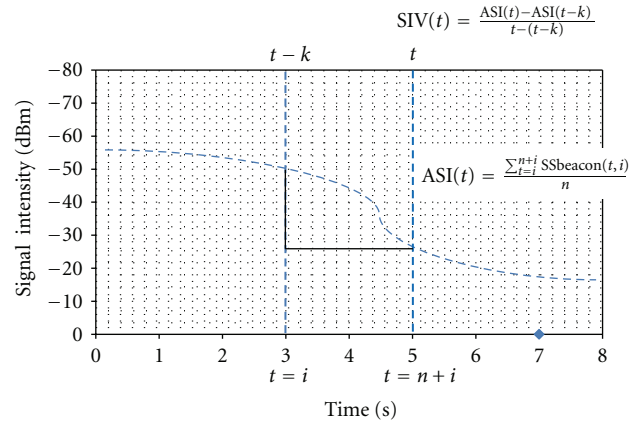


Figure 8: FLPC input parameters.

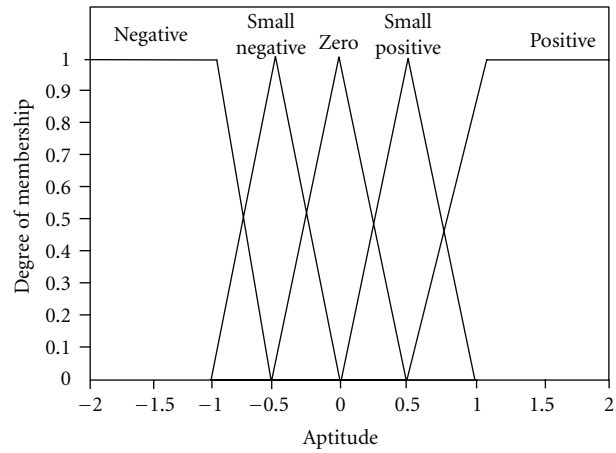


Figure 9: Output variable: aptitude value.

Table 2: Fuzzy system rules matrix.

ASI	SIV		
	Positive	Zero	Negative
Excellent	Positive	Positive	Small positive
Good	Positive	Small positive	Zero
Medium	Small positive	Zero	Small negative
Low	Zero	Small negative	Negative

(12) If ASI is low and SIV is negative then aptitude is negative.

The number of linguistic terms of each entry sets the number of rules, in this case $4 \times 3 = 12$ rules. The knowledge base arrangement is illustrated by the matrix shown in Table 2. The decision taken by the predictive fuzzy control is clearly defined by the two inputs values. Table 3 provides a synthesis of the values that define the membership functions.

3.2. Fuzzy Logic Implementations. Hardware implementation was carried on using the "SmartFusion Evaluation" development kit Actel FPGA A2F200M3F (Figure 10), whose main specs are as follows: 200 K gates, 256 Kb memory

Table 3: Parameters of membership functions.

Linguistic variable	Linguistic term	Type of function	Parameters			
Average signal intensity	Low	Trapezoidal	−80	−80	−70	−50
	Medium	Triangular	−60	−50	−40	
	Good	Triangular	−50	−40	−30	
	Excellent	Trapezoidal	−40	−30	−20	−20
Signal intensity variation	Negative	Trapezoidal	−3	−3	−2	0
	Zero	Triangular	−1	0	1	
	Positive	Trapezoidal	0	1	3	3
Aptitude	Negative	Trapezoidal	−2	−2	−1	−0.5
	Small Negative	Triangular	−1	−0.5	0	
	Zero	Triangular	−0.5	0	0.5	
	Small Positive	Triangular	0	0.5	1	
	Positive	Trapezoidal	0.5	1	2	2

flash, 64 Kb SRAM, 20 Mhz crystal oscillator [26], and 32 bits ARM Cortex M3 soft processor (Figure 11). The FPGA using its logical blocks, allows customization through the "Libero-" integrated design software, firmware edition, and compilation in C language.

Fuzzytech software [27] was used to design the FIS. The PFLC was set to 16 bits resolution and was evaluated using the C-function *EvalFis(float Var_ASI, float Var_SIV)*, which receives the input values ASI and SIV and returns the result of the output variable Aptitude. Figure 11 illustrates the embedded FLPC design. We used the C language tools of Fuzzytech to develop the algorithm of the PFLC.

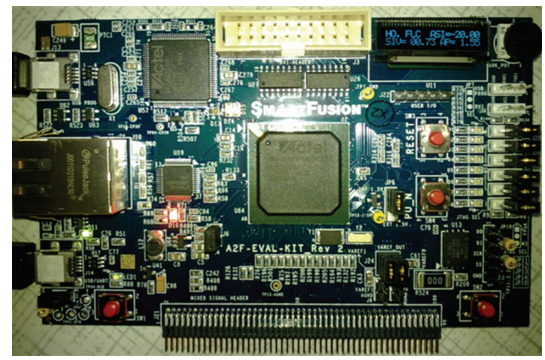


Figure 10: Actel development board for FPGA.

4. Experiments and Results

This section presents some of the tests performed in order to validate the PFLC design and performance.

4.1. Itinerant MS in Source Cell: No Handoff. In its simplest expression, an MS moves around within the limits of its serving cell. As usual, beacon frames are received at 100 msec. intervals and used to calculate ASI. SIV is updated every 2 sec., as well as the aptitude value. Since a user can find obstacles and move around to avoid them, signal intensity by itself is not a good reference to determine an MS's direction. So, the SIV provides this information, as stated before in (2).

Table 4 shows some sample values of the AP's signal intensity received by the MS.

Table 4: Intensity values received by the MS.

Signal intensity (dBm)	Sequence of beacons	Time in (msec.)
−79.12	1	100
−78.73	2	200
−78.52	3	300
−77.92	4	400
−77.40	5	500
...	...	...
−78.00	20	2000

The first value is close to the minimum detectable signal strength, indicating that the MS is far from the AP and near a boundary. As the MS moves, the values start increasing, which implies that the MS is getting closer to the AP. Table 5 shows various ASI values obtained.

As illustrated in Figure 11, the PFLC performs its decisions based on the two inputs (ASI, SIV) and its knowledge base, yielding a numerical result, known as aptitude value. In our design, +3 is the maximum aptitude, and −3 the minimum. Figure 12 presents an instance of calculation.

4.2. Itinerant MS Crossing Cell Boundaries in an Eleven Aps Ensemble. The simulation considers an MS receiving the beacon frames from a group of eleven surrounding cells. To illustrate cell crossings and related handoff, Figure 13 shows four cells only.

The MS's FLPC performs ASI-SIV calculations every two seconds to obtain the aptitude value for eleven APs and select the one with the higher value; in this way, the ARM processor acquires the values of the eleven ASI and SIV signals, then the embedded FLPC executes each inference and stores each APs output result, then selects the maximum value.

When the proper conditions are met to trigger a handoff, the best positioned AP and channel are known. As the MS travels along the trajectory, it navigates through different coverage areas, aptitude values are computed, and beacon signals from various cells become significant and competitive

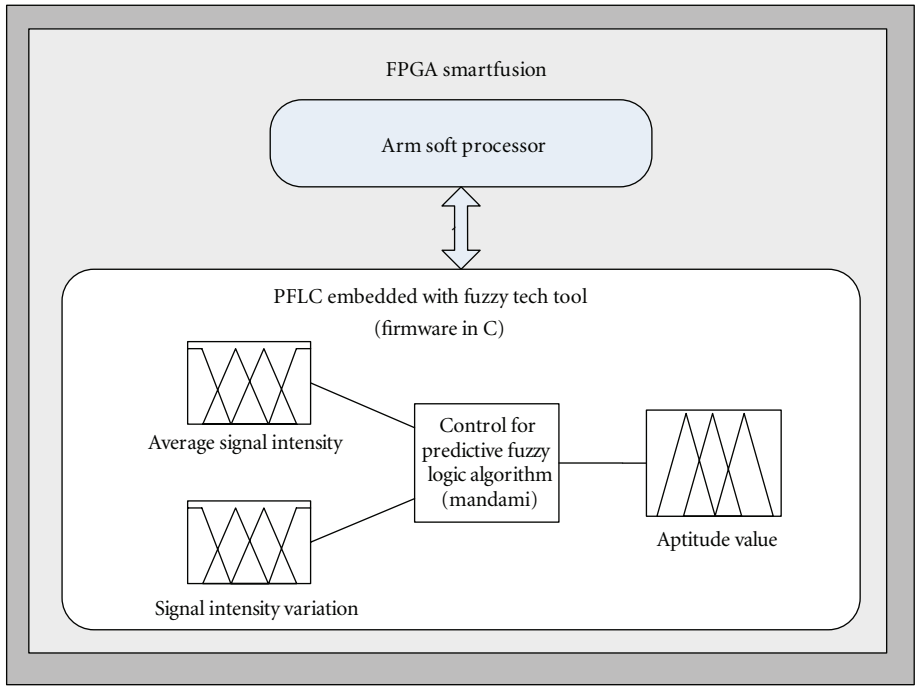


Figure 11: Fuzzy system embedded in FPGA smartfusion.

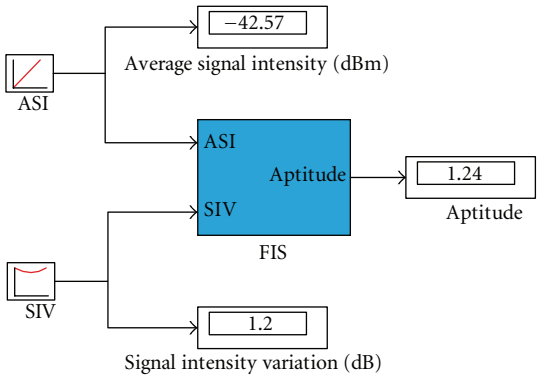


Figure 12: Aptitude value for the source AP.

Table 5: ASI values.

ASI (dBm)	Time (sec.)
−78.44	2
−76.55	4
−74.54	6
−72.54	8
...	...
−20	n

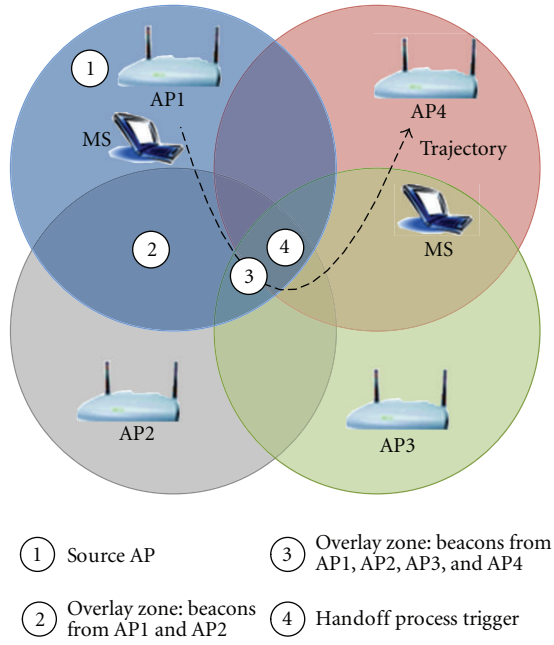


Figure 13: Multiple APs.

versus the source's, but it is until the MS gets to point 4, when the conditions are set to trigger a Handoff. Figure 14 illustrates the PFLC embedded for eleven channels, with the selection of a best aptitude.

Under normal conditions, IEEE 802.11 proceeds to a scanning process of eleven channels to determine the one with the highest signal strength, and then decide which one will be the target AP. In our proposal, the scanning process is reduced to a single channel, since the MS knows in advance which AP has the higher aptitude.

Figure 15 shows the three-dimensional control surface computed by the Mamdani-type FLPC. The axes values correspond to the ASI, SIV, and Aptitude. As can be

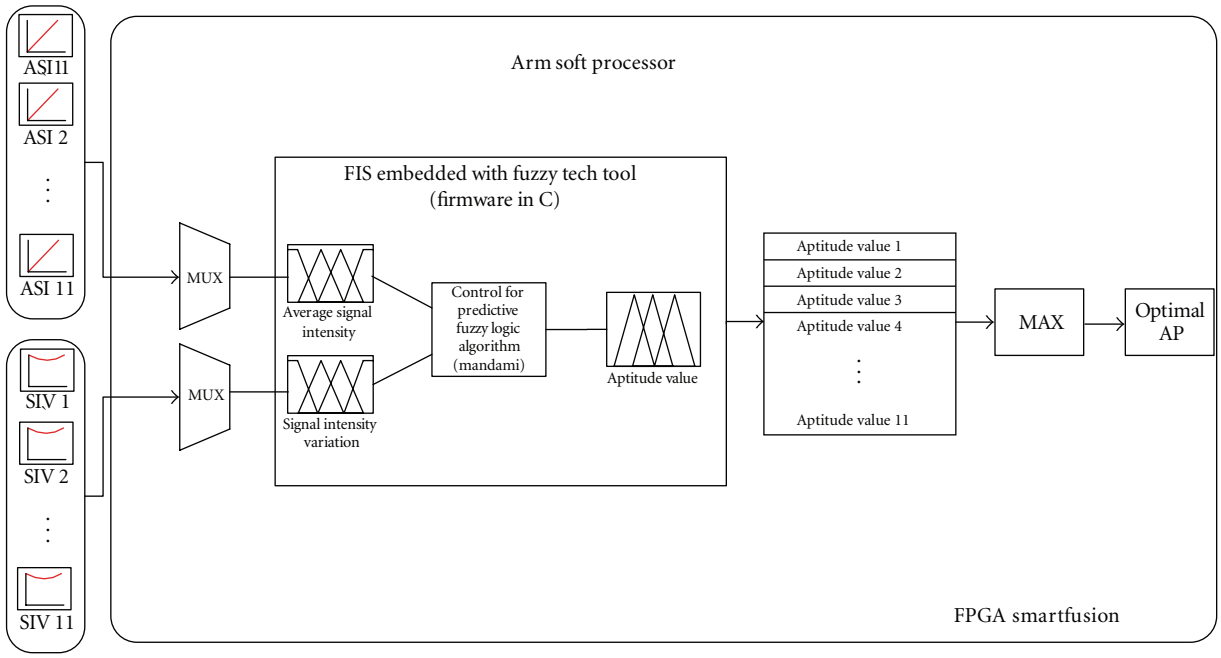


Figure 14: Fuzzy control in operation with multiple APs.

Table 6: Comparison of the output using the fuzzy toolbox of Matlab and the FLPC in the FPGA.

	Input		Output	
	ASI	SIV	Matlab	FLPC in the FPGA
1	−23.27	1.03	1.38	1.50
2	−54.22	−1.34	−0.85	−0.76
3	−35.10	0.85	1.22	1.28
4	−42.77	−2.86	−0.16	−0.14
5	−60.35	−0.25	−0.75	−0.69
6	−71.97	0.36	−0.31	−0.32
7	−56.13	2.83	0.28	0.27
8	−44.30	−2.68	−0.22	−021
9	−80.00	−0.10	−0.60	−0.55
10	−20.21	0.73	1.35	1.50

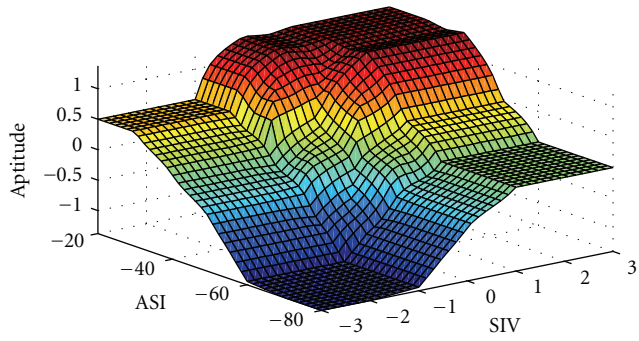


Figure 15: Fuzzy system control surface.

seen, when SIV is negative and ASI is minimum, aptitude values are negative, indicating an AP which should not be considered as the destination for a handoff. The highest values of Aptitude ($1 < z \leq 2$) are obtained from the conjunction of high ASI values ($-30 < x \leq -20$ dBm) and positive SIV values ($2 < y \leq 3$).

Figure 15 shows the control surface of the fuzzy handoff controller. This surface controls the rate of change of the AVS and SIV signals and it was obtained experimentally; however it can be changed using any method such as the simple tuning method [28], or considering other objectives, in this case it will be necessary to use a multiobjective optimization method in order to adapt the membership functions [29].

For a comparative analysis, in Table 6 we have the output for different values of ASI and SIV, of the PFLC designed with the fuzzy toolbox of Matlab [30], and the value obtained with our proposal. The differences are minimum.

5. Conclusions

VoIP is a real time application sensitive to time delays over 200 msec. Standard IEEE 802.11 can have handoff delays of up to 300 msec. The capability to maintain uninterrupted VoIP connections while crossing through multiple WiFi cells is fundamental for this communication system's survival in the contested wireless arena. A predictive fuzzy logic control based on average signal intensity and signal intensity variation parameters as inputs to the fuzzy inference algorithm yields access point's aptitude value for up to eleven APs, selects the highest, and designates the best AP to perform a cell handoff. This approach reduces the handoff scanning time nearly 30 msec, and the complete HO to around 60 msec., thus providing better quality of service to

VoIP and other real time applications, where multiple-cell arrangements are involved.

Soft computing tools (fuzzy logic, neural networks, and genetic algorithms) usage is on the rise to tackle highly complex problems, yielding for the most part satisfactory outcomes. Fuzzy logic has been widely used in control and prediction applications because of its human language parallelism, that makes it independent of rigorous analysis precision, thus providing analysis capabilities in uncertainty situations, where not-well-defined limits are found. This is the case in the decision-making process of the MS to launch a handoff while nearing its serving cell boundary. In a multiple-cell environment, received power and direction are computed in milliseconds intervals for the inference engine to decide the time and best fitted target AP.

Nowadays the Systems-on-Chip (SoC) are highly dense silicon chips, which integrate complete customized systems comprised of microprocessors and I/O devices to interface screens and keyboards, as is the case of cellular terminals. Therefore, the inclusion of a real-time high-performance fuzzy systems allows to achieve multiple tasks without sacrificing performance.

Acknowledgment

The authors would like to thank the Instituto Politécnico Nacional (IPN), Comisión de Operación y Fomento de Actividades Académicas (COFAA), and the Mexican Consejo Nacional de Ciencia y Tecnología (CONACYT) for supporting their research activities.

References

[1] *Wi-Fi Alliance*, Press Release, Austin, Tex, USA, 2012.

[2] Institute of Electrical and Electronics Engineers, "IEEE Standards Association, 802.11: Wireless LANs".

[3] LTE, *WiMAX and WLAN Network Design, Optimization and Performance Analysis Leonhard Korowajczuk*, John Wiley & Sons, 2011.

[4] I. Purushothaman and S. Roy, "FastScan: A handoff scheme for voice over IEEE 802.11 WLANs," *Wireless Networks*, vol. 16, no. 7, pp. 2049–2063, 2010.

[5] ITU-TG.114, *One-Way Transmission Time*, 2003.

[6] Y. Maldonado, O. Montiel, R. Sepúlveda, and O. Castillo, "Design and simulation of the fuzzification stage through the Xilinx system generator," *Studies in Computational Intelligence*, vol. 154, pp. 297–305, 2008.

[7] J. A. Olivas, R. Sepúlveda, O. Montiel, and O. Castillo, "Methodology to test and validate a VHDL inference engine through the Xilinx system generator," *Studies in Computational Intelligence*, vol. 154, pp. 325–331, 2008.

[8] G. Lizárraga, R. Sepúlveda, O. Montiel, and O. Castillo, "Modeling and simulation of the defuzzification stage using Xilinx system generator and simulink," *Studies in Computational Intelligence*, vol. 154, pp. 333–343, 2008.

[9] S. Poorony, T. V. S. Urmila Priya, K. Udaya Kumara, and S. Renganarayanan, "FPGA based fuzzy logic controller for electric vehicle," *Journal of the Institution of Engineers*, vol. 45, no. 5, pp. 1–14, 2005.

[10] O. Montiel, R. Sepúlveda, Y. Maldonado, and O. Castillo, "Design and simulation of the type-2 fuzzification stage: Using active membership functions," *Studies in Computational Intelligence*, vol. 257, pp. 273–293, 2009.

[11] R. Sepúlveda, O. Montiel, J. Olivas, and O. Castillo, "Methodology to test and validate a VHDL inference engine of a type-2 FIS, through the xilinx system generator," *Studies in Computational Intelligence*, vol. 257, pp. 295–308, 2009.

[12] R. Sepúlveda, O. Montiel, G. Lizárraga, and O. Castillo, "Modeling and simulation of the defuzzification stage of a type-2 fuzzy controller using the xilinx system generator and simulink," *Studies in Computational Intelligence*, vol. 257, pp. 309–325, 2009.

[13] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Modelling and simulation of the defuzzification stage of a type-2 fuzzy controller using VHDL code," *Control and Intelligent Systems*, vol. 39, no. 1, pp. 33–40, 2011.

[14] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Embedding a high speed interval type-2 fuzzy controller for a real plant into an FPGA," *Applied Soft Computing Journal*, vol. 12, no. 3, pp. 988–998, 2012.

[15] L. C. Godara, *Handbook of Antennas in Wireless Communications*, CRC Press, 2002.

[16] "Certified Wireless Network Administrator TM: Official Study Guide, Planet3 Wireless," 2002.

[17] A. R. Prasad and N. R. Prasad, *802.11 WLANs and IP Networking*, Artech House, 2005.

[18] A. Mishra, M. Shin, Arbaugh, and W, "An empirical analysis of the IEEE 802.11MAC layer handoff process," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 93–102, 2003.

[19] S. Shin, A. G. Forte, A. S. Rawat, and H. Schulzrinne, "Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs," in *Proceedings of the 2nd International Workshop on Mobility Management and Wireless Access Protocols (MobiWac '04)*, pp. 19–26, October 2004.

[20] C. S. Li, Y. C. Tseng, and H. C. Chao, "A neighbor caching mechanism for handoff in IEEE 802.11 wireless networks," in *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE '07)*, pp. 48–53, April 2007.

[21] C. Y. Chang, H. J. Wang, H. C. Chao, and J. H. Park, "IEEE 802.11 handoff latency improvement using Fuzzy Logic," *Wireless Communications and Mobile Computing*, vol. 8, no. 9, pp. 1201–1213, 2008.

[22] Y. Song, M. Liu, Z. Li, and Q. Li, "Handover latency of predictive FMIPv6 in IEEE 802.11 WLANs: a cross layer perspective," in *Proceedings of the 18th International Conference on Computer Communications and Networks (ICCCN '09)*, Honolulu, Hawaii, USA, August 2009.

[23] E. H. Ong and J. Y. Khan, "QoS provisioning for VoIP over wireless local area networks," in *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS '08)*, pp. 906–911, November 2008.

[24] C. G. Patil and M. T. Kolte, "An approach for optimization of handoff algorithm using fuzzy logic system," *International Journal of Computer Science and Communication*, vol. 2, pp. 113–118, 2011.

[25] A. M. Gharehbaghi and M. A. Badamchizadeh, "An improved method for fuzzy congestion control," *International Journal of Computer and Network Security*, vol. 2, no. 7, pp. 1–5, 2010.

[26] "Microsemi-Actel Corporation: SmartFusion Evaluation Kit User's Guide," USA, 2012.

[27] "Fuzzy tech: fuzzyTECH 5.5 user's manual, INFORM GmbH /Inform Software Corp," 2001.

[28] O. Montiel, R. Sepúlveda, P. Melin, O. Castillo, M. Á. Porta, and I. M. Meza, "Performance of a simple tuned fuzzy controller and a PID controller on a DC motor," in *Proceedings*

of the IEEE Symposium on Foundations of Computational Intelligence (FOCI '07), pp. 531–537, April 2007.

[29] Y. Maldonado, O. Castillo, and P. Melin, "Optimization of membership functions for an incremental fuzzy PD control based on genetic algorithms," *Studies in Computational Intelligence*, vol. 318, pp. 195–211, 2010.

[30] R. Sepúlveda, O. Montiel-Ross, C. Juan Manzanarez, and E. Ernesto Quiroz, "Fuzzy logic predictive algorithm for wireless-LAN fast inter-cell handoff," *Engineer Letters*, vol. 20, no. 1, pp. 109–115, 2012.

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 475894, 11 pages
doi:10.1155/2012/475894

Research Article

Designing High-Performance Fuzzy Controllers Combining IP Cores and Soft Processors

Oscar Montiel-Ross, Jorge Quiñones, and Roberto Sepúlveda

Instituto Politécnico Nacional, CITEDI, Avenida del Parque 1310, 22510 Tijuana, B.C., Mexico

Correspondence should be addressed to Oscar Montiel-Ross, o.montiel@ieee.org

Received 5 May 2012; Accepted 3 June 2012

Academic Editor: Oscar Castillo

This paper presents a methodology to integrate a fuzzy coprocessor described in VHDL (VHSIC Hardware Description Language) to a soft processor embedded into an FPGA, which increases the throughput of the whole system, since the controller uses parallelism at the circuitry level for high-speed-demanding applications, the rest of the application can be written in C/C++. We used the ARM 32-bit soft processor, which allows sequential and parallel programming. The FLC coprocessor incorporates a tuning method that allows to manipulate the system response. We show experimental results using a fuzzy PD+I controller as the embedded coprocessor.

1. Introduction

Nowadays, the term System on Chip (SoC) gains terrain since the trend towards the use of a highly efficient hardware platform for real-time processing is increasing [1]. Modern FPGA devices that allow to mix digital and analog signal makes them a good choice to use them for SoC design [2].

FPGA platforms are designed to achieve higher integration levels in low-power low-cost electronic systems by embedding processors with efficient architectures such as DSP, RISC, multicore processor systems, buses, on-chip memory blocks, peripheral devices [3]. FPGA-based system architectures support the combination of user defined synthesizable Intellectual Property (IP) blocks with IP blocks and software drivers and libraries provided by the manufacturer [4]. The use of SoC allows total independence of a desktop computer system, reducing costs, and increasing performance of applications, avoiding communication interface bottlenecks, latency, and data loss [5].

The SoC design based on modern FPGAs may incorporate dedicated processors embedded in the silicon known as "Hard", and programmable processors known as "Soft" that are implemented in the programmable logic resources of the FPGA, or a mixed of both.

The real-world problems are diverse and complex, treating many of them becomes difficult, because they make us face many scientific and technological barriers, such as, mathematical modeling and high-speed processing for data processing and control applications [6].

For control applications, there exist many techniques and strategies to make a system behaves according to a plan; for example, to use a Fuzzy Logic Controller (FLC) that is considered as a control strategy based on rules, which are usually raised by the knowledge from an expert. This may be crucial in control problems that could present difficulties in constructing accurate mathematical models [7, 8].

There are two typical options to carry out applications of FLC embedded into an FPGA.

(1) To describe the FLC in C language and then use a specialized tool (compiler) to translate Handel C to bitstream [9, 10].

(2) To use VHDL to describe the FLC, this can be used as (a) standalone controller, (b) incorporated to a hard/soft processor as a Soft Core connected to the system bus, (c) incorporated to a hard/soft processor through an internal input/output interface [11, 12].

The complexity of modern systems requires more design efforts by increasing developing costs; so, to minimize them, the reutilization of already tested circuits is a necessity. Any functional component to be reused in the form of

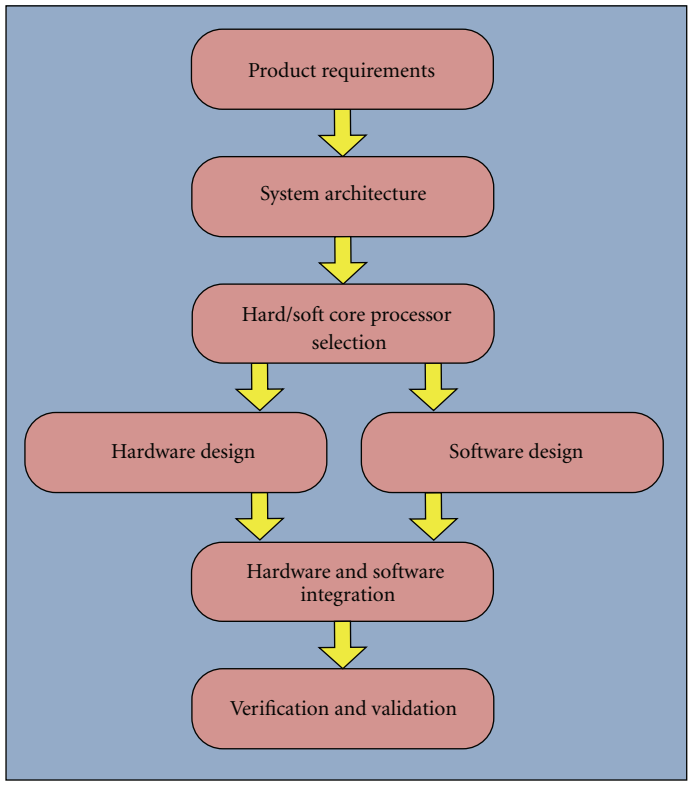


Figure 1: Design process of an embedded system.

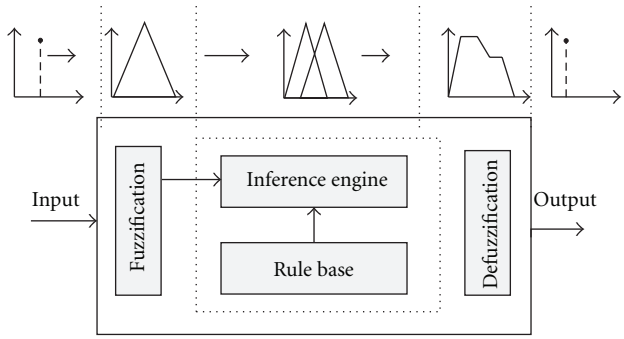


Figure 2: Fuzzy inference system.

an already-designed electronic component, or fabricated hardware constitute an Intellectual Property Core (IP-Core).

This paper presents a methodology to integrate a FLC to a SoC through an input/output internal port; the idea is to use the FLC as a coprocessor. Furthermore, an experimental study of high-performance computing using fuzzy modeling implemented in FPGA-based systems is presented.

Other studies that have addressed the same subject with a different focus are. In [13], the authors describe their experiences designing real-time hardware/software for SoC and they addressed the problem of data acquisition for the ANTARES neutrino experiment, and for the problem of a selective read-out processor for an electromagnetic calorimeter. In [14, 15], the authors present the architecture and VHDL code to implement a type-1 FLC, while in [16] the architecture and VHDL code to implement a type-2 FLC and experiments are presented.

In literature, there exist many interesting works that deal with fuzzy controllers that can be embedded into an FPGA, for example. [17] presents the genetic optimization of Membership Functions (MFs) for an Incremental Fuzzy PD Controller, [18] shows the optimization of MFs to regulate a servomechanism with backlash.

The organization of this paper is as follows. Section 2 explains the design process of an embedded system, how to integrate a fuzzy coprocessor into a SoC, and the debugging process. In Section 3, the problem formulation, experimental plant, control objective, and electromechanical limitations concerning time are given. In Section 4, the generic architecture that integrates the ARM processor and the FLC as Intellectual Property cores (IP core) is described. In Section 5, the main characteristics of the FLC are described, and the methodology to tune the FLC using one variable is explained. In Section 6, the experiments' sets and results are explained. Finally, in Section 7, the conclusions of this work are given.

2. Integrating a Fuzzy Coprocessor in a SoC

Figure 1 illustrates the design process of a SoC-embedded system. It is a standalone dedicated hardware computer with custom peripherals incorporated to the system as IP cores, and specialized software to solve a specific problem.

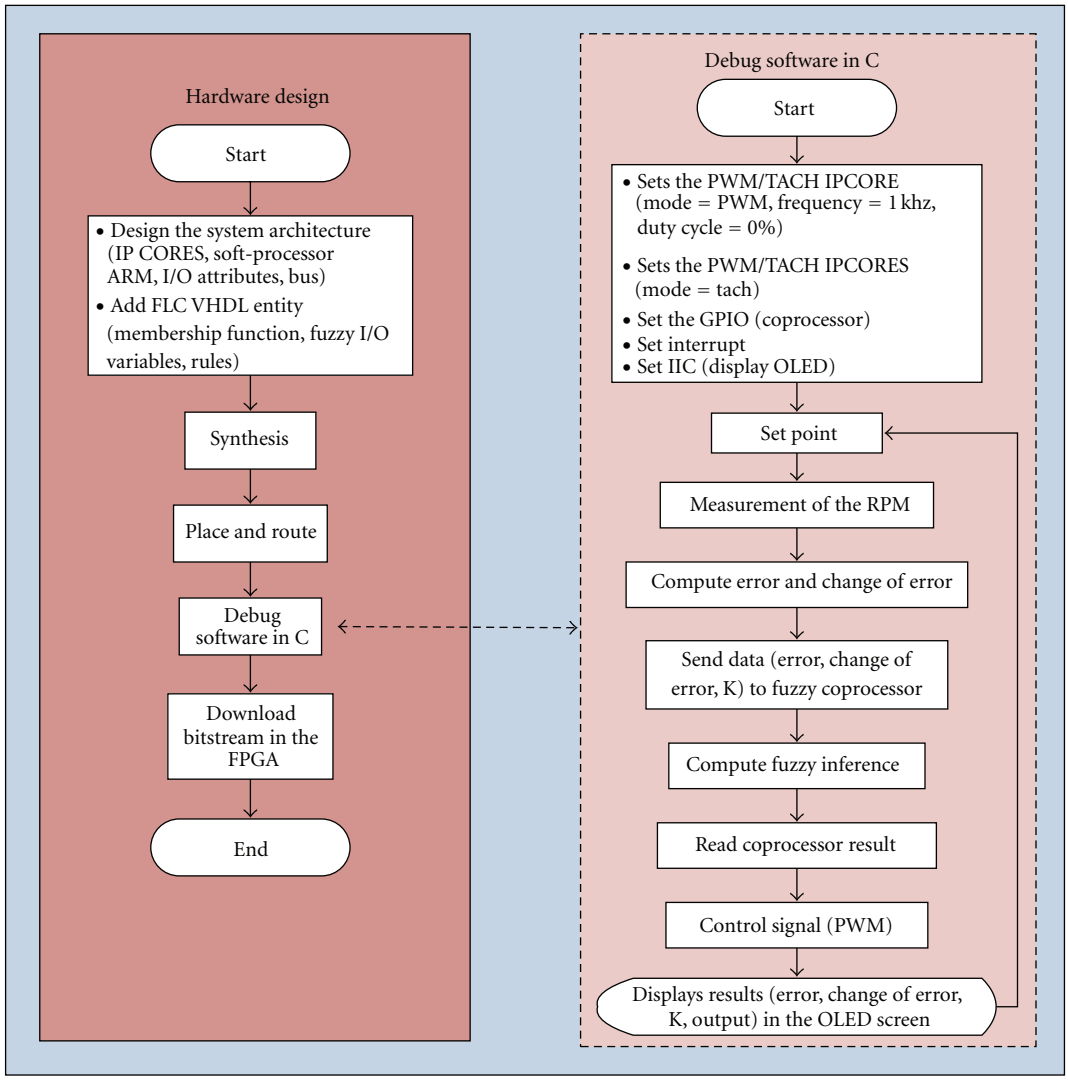


Figure 3: Flow diagram of the debugging process of a C program executed in the FPGA base system.

2.1. Designing the FLC as IP Core. Figure 2 shows the typical way of representing a FLC; it is composed of three stages: the fuzzification stage, the inference engine that contains the rule base and database, and defuzzification stage [19].

In general, the steps to incorporate into an FPGA Fusion an FLC as Soft IP core using the Libero software are.

(1) Create the Entity Design of the FLC by using either a Hardware Description Language (HDL), structural schematic, or mixed-mode (schematic and Register Transfer Logic "RTL"). We used VHDL (Very High Speed Integrated Circuit HDL) to specify all the fuzzy engine and parameters.

(2) Test the functionality of the FLC Entity Design using a test bench program such as the Model Sim for VHDL.

(3) Create the soft processor and incorporate the FLC IP coprocessor [20] entity to the system bus trough the GPIO IP.

(4) Edit and debug the ARM firmware in C language to complement the controller's design incorporating the reading and conditioning of the process variables.

(5) Download the bitstream to the target FPGA development board.

The obtained FLC is a portable, reusable, and nonencrypted Soft IP core. It was designed to be used through an input/output interface, which is an advantage because many of the commercial IP cores have the inconvenient that are not compatible with all the system buses.

Figure 3 illustrates the hardware design and debugging of the software written in C language.

3. Problem Formulation

The main target of this work is to give a methodology to integrate a soft IP FLC to a SoC in order to develop high-performance applications; whereas, the experiments

Figure 4: Inverted pendulum system.

are focused to demonstrate the advantages of using high-performance FLC for real-life problems. To achieve the objective paper, we are including two well-known problems.

In the first problem, we used a plant with the next main components: a Pittman DC geared-motor model GM9236S025-R1, a ± 12 V_{DC} power Supply, an H-bridge board, and a PWM system to provide the motor with the necessary average power to reach the desired speed with and without load, and disturbances. Therefore, the *control objective* is to reach the wanted speed as fast as possible with the minimal overshoot, which is achieved by the correct calculation of the relation T_{ON}/T_{OFF} to fulfill the target.

Two important characteristics of this motor are the electrical time constant τ_E and the mechanical time constant τ_M whose values are 1.06 ms and 8.5 ms, respectively; so, we can consider global response time constant of 9.16 ms. This time is significant since it is the time required for the motor's speed to attain 63.2% of its final value for a fixed voltage level.

The DC motor alone is a linear plant since the no-load speed is directly proportional to the DC supply voltage. However, the system becomes nonlinear when a switching supply is applied, in this case by using PWM; in addition, the application of load and disturbances increases nonlinearities.

In the second problem, the plant is an inverted pendulum; see Figure 4. The cart is mounted over an aluminum frame; it is moved from one side to the other using a ball screw controlled by a motor. The pole balancing is mounted over the cart and coupled to a quadrature optical encoder Model 121, which provides 300 counts per revolution (CPR).

4. SoC General Description

In Figure 5, the designed system architecture for FPGA Fusion [21] of the Actel company is shown, embedded into the FPGA are. The ARM processor, two memory blocks, a general-purpose input output (GPIO) interface, timers, interrupt controller (IRQ), IIC, serial port (UART), pulse width modulator/tachometer block, and the FLC block [14, 15]; all the embedded components are IP cores. External to the FPGA are a DC motor with a high-resolution quadrature optical encoder, the plant's power supply, an H-bridge for power control, a personal computer, and a digital display.

The FPGA Fusion allows to incorporate the soft processor ARM cortex, as well as other IP Cores (IPCORE) to make a custom configuration. The ARM cortex handles a 32-bit bus for peripheral control named Advanced Peripheral Bus (APB). Note in Figure 5 that the block FLC contains the design of the fuzzy controller integrated to the system as a soft IP CORE, similarly the rest of the IPCOREs are general purpose devices from the catalog of cores provided by Actel and the system board seller. In this system, the FLC IP core has the advantage that it is not dependable of the bus system, and it provides to the user the capacity of handling very easy without the need of knowing the internal functionality; on the other hand, because it is not encrypted, the content can be read.

Figure 5 shows how to connect the FLC IP core to the ARM processor through the GPIO. The FLC has seven inputs and two output. The inputs are "clk", "ce", "rst", "k", "error", "c.error", and "w". The input "clk" is a 50 Mhz system clock, the aim of the "ce" input is to enable or disable the FLC, and the input "rst" restores all the internal registers of the FLC, the input "w" working together with "ce" allows to start a fuzzy inference cycle; all they are one-bit size. The four-bit input "k" is for modifying the support of the input membership functions to change the system response. The eight-bit input "Error" and Change of Error "c.error" are the controller inputs. The outputs are "Out" and "IRQ". "Out" is eight-bits wide and it is the crisp output value. "IRQ" is the interrupt request FLC output. The system has five membership functions (MFs) for each input and output. The input and output MFs are trapezoidal at the extremes, and the remaining are triangular. This FLC uses the simple tuning algorithm for modifying the system's response in an intuitive way [14, 15]. Figure 6 shows the FLC entity and GPIO IP bus connection.

The GPIO IP has two 32-bit wide ports; one for output (write bus) and one for input (reading bus). The output bus connects the GPIO IP to the ARM cortex using the 32-bit bus APB. The input bus connects the FLC IP to the GPIO IP. In this configuration, the FLC works as a coprocessor of the ARM cortex processor.

The ARM processor writes a "k(3:0)" value for setting up the input MFs, it provides the Error and Change of error values for the "Error(7:0)" and "c.error(7:0)" FLC

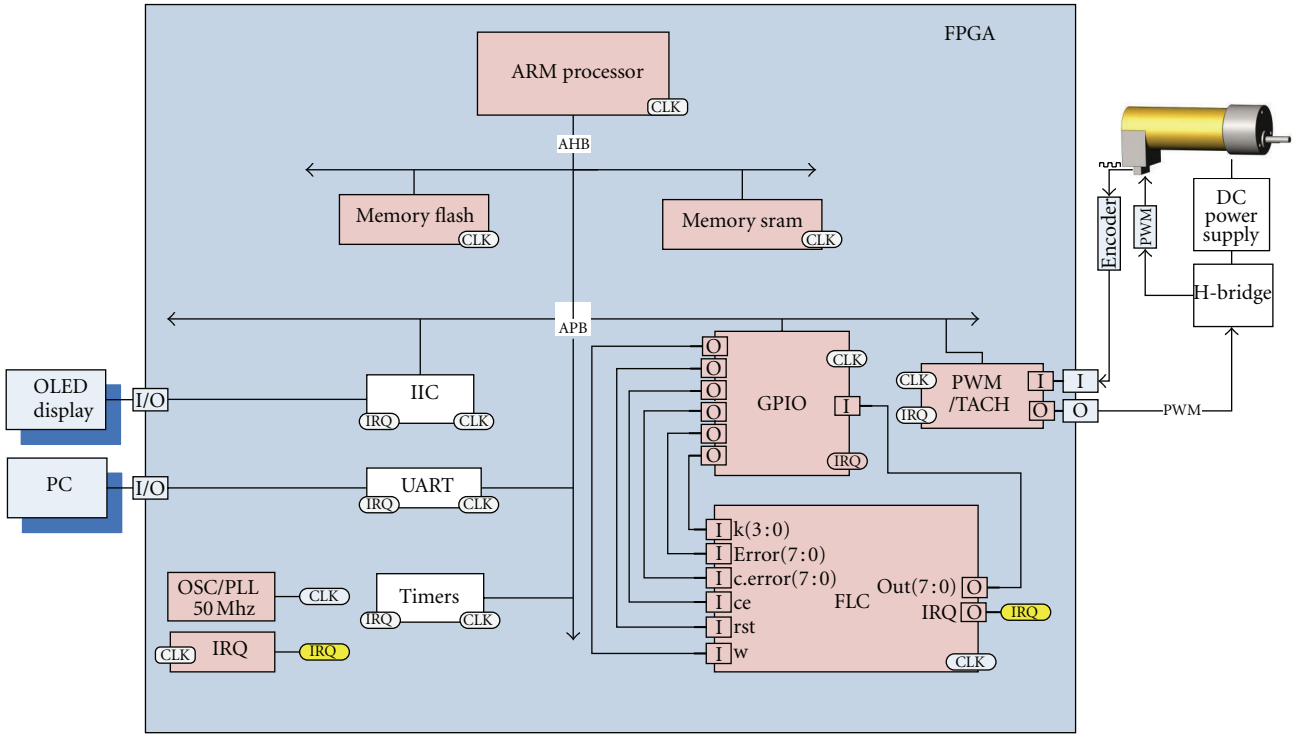
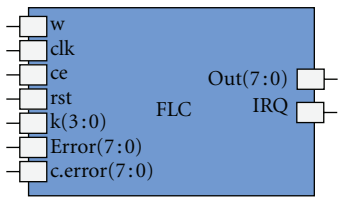


Figure 5: Overview of the general system.



Output bus																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not connected									w	ce	rst	c.error								Error								k			

Input bus																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Ground																								Out							

Figure 6: FLC entity and GPIO bus connection.

inputs, this is achieved using the output variables k[3:0], Error[11:4], and c.error [19:12] of the GPIO IP. Similarly, the ARM reads the FLC output "Out(7:0)" through the GPIO IP using the 8 bit input variable Out(0:7).

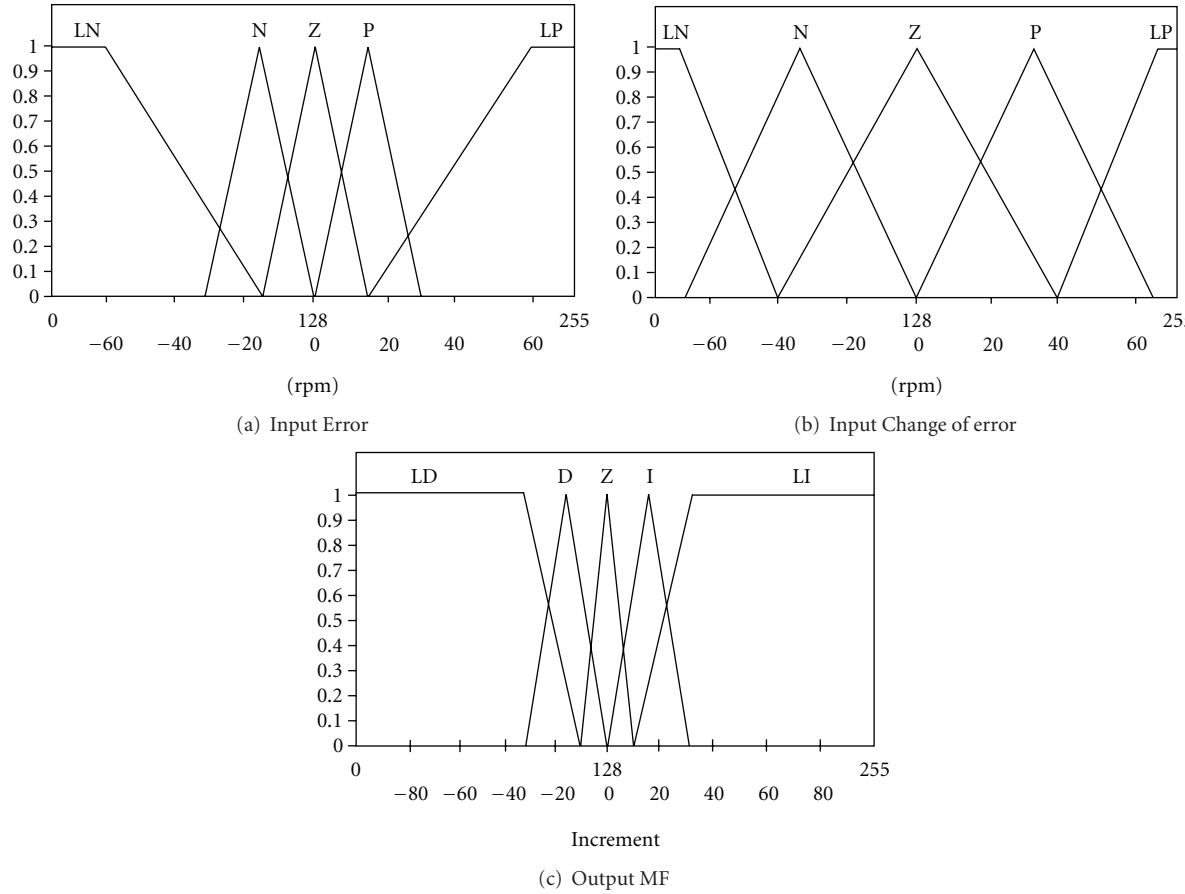
5. FLC Characteristics

For each input of the FLC, Error and Change of error, we defined five MFs: LN (large negative), N (negative), Z (zero), P (positive), and LP (large positive). The universe of discourse for these membership functions is in the range $[-80, 80]$.

For the output of the FLC, we have five MFs: LD (large decrement), D (decrement), Z (zero), I (increment) and LI (large increment), with the universe of discourse in the interval $[-100, 100]$.

Figure 7 shows the input/output membership functions, and the rule matrix of the FLC, integrated by 25 rules, is shown in Figure 8.

5.1. Tuning the Controller. The FLC IP has a three-bit input to manipulate the MFs to improve the desired response by using the method referred in literature as Simple Tuning Algorithm (STA) [22, 23]. There are different methods to tune a FLC, for example, evolutionary computation, Fuzzy Knowledge Base Controllers, and so forth, where the aim is to search the optimal solution in base of objective function, gradient error, and others but in most of the cases these



(a) Input Error

(b) Input Change of error

(c) Output MF

Figure 7: The FLC has two inputs: Error and Change of error, and one output. The output provides a value of increment/decrement to be use by the integral part of the PD+I controller. The first scale $[0, 255]$ are the real MFs bit values definition in the FLC. The second scale $[-80, 80]$ and $[-100, 100]$ are the operation values for this application. The scale conversion is achieved by the ARM processor.

ce \ e	LN	N	Z	P	LP
LN	LI	LI	I	D	LD
N	LI	I	Z	Z	LD
Z	LI	I	Z	D	LD
P	LI	Z	Z	D	LD
LP	LI	I	D	LD	LD

Figure 8: Rule matrix of the FLC.

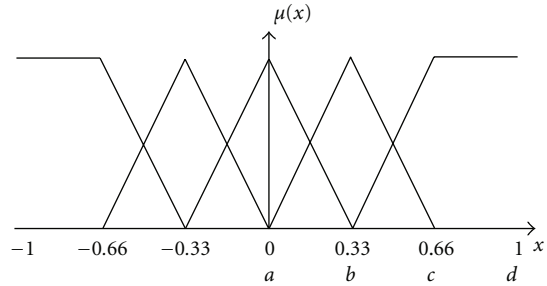


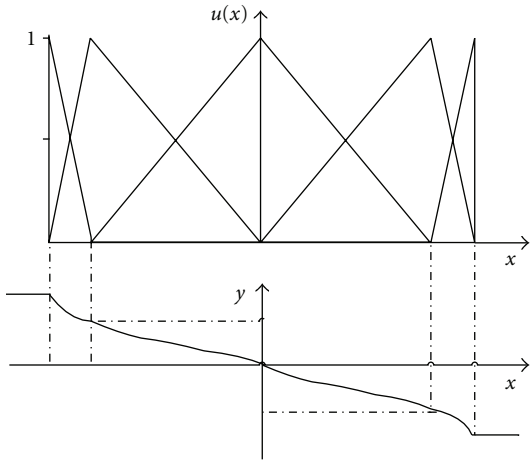
Figure 9: Normalization of the input MFs. The tuning factor is $r = 1$.

methodologies have convergence and mathematical representation problems; furthermore, often they require many system resources and have considerable big computational complexity for online adaptation of real time systems.

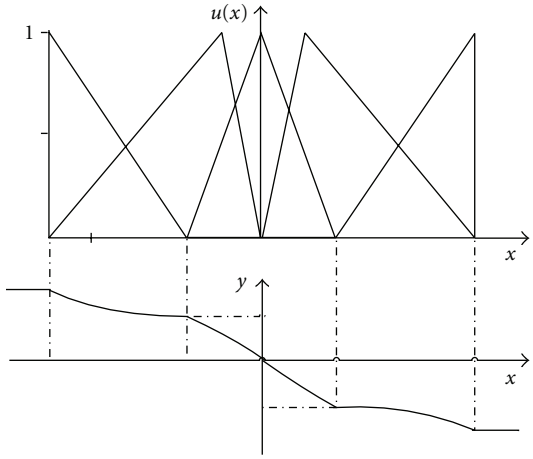
The STA takes advantage of the typical characteristic of error behavior around the set point, so it is possible to use a predefined set of MFs and rules and modify their support according to a very simple arithmetic expression. In this example, the two inputs were modified using only a tuning factor [23]; however, it is likely to modify both inputs using two tuning factors [22]. In Figure 9, the normalized inputs are shown; Figures 10(a) and 10(b) show how the MFs are modified by the application of the STA algorithm and their effect in the system behavior.

Considering all the points in the universe of discourse that defines every MF of each input, a vector named $V_{\mathrm{OP_{initial}}}$, is codified. The basic idea of the STA is to change this vector using an exponential function $r(k)$ named the tuning factor such as $V_{\mathrm{OP_{final}}} = (V_{\mathrm{OP_{initial}}})^{r(k)}$, that produces the changes to the system which is shown in Figure 10. This leads us to analyze the next three cases using Figure 9:

(1) $r = 1$: this is the case where the normalized MFs have no change; that is, $a - b = b - c = c - d$,



(a) The tuning factor is $r < 1$



(b) The tuning factor is $r > 1$

FIGURE 10: Modifying the input MFs using the formula $V_{\mathrm{OP}_{\mathrm{final}}} = (V_{\mathrm{OP}_{\mathrm{initial}}})^{r(k)}$.

(2) $r < 1$: after the application of the tuning factor, the distances between the middle MFs operation points are larger than the external; so, $a^r - b^r > c^r - d^r$, producing the expansion of the MFs, as illustrated in Figure 10(a).

(3) $r > 1$: the opposite case of previous, after the application of the tuning factor, the distances between the two external MFs are larger than the others; that is, $a^r - b^r < c^r - d^r$. This produces the compression of the MFs, this effect is shown in Figure 10(b).

Therefore, the STA consists basically in four steps.

(1) *Tuning Factor Selection.* A number $k \in [0, 1]$ to define the tuning adjustment level is used. $k = 0$ is the biggest time, and $k = 1$ the smallest.

(2) *Normalization of the Ranges of the Fuzzy Controller's variables.* The range of each input fuzzy variable is modified in order to have the lower and upper limits equal to -1 and $+1$, respectively, see Figure 9.

(3) *Tuning Factor Processing.* Once the range is normalized, the new vector of operation points will be given by.

$$V_{\mathrm{OP}_{\mathrm{final}}} = (V_{\mathrm{OP}_{\mathrm{initial}}})^{r(k)}, \tag{1}$$

where $V_{\mathrm{OP}_{\mathrm{initial}}}$ are the normalized values of the MFs in the x-axis and $r(k)$ can be one of the following polynomials:

(a) The value $r \in [1/40, 3]$

$$r(k) = \frac{30k^3 + 37k^2 + 52k + 1}{40}, \tag{2}$$

(b) The value $r \in [0, 4]$

$$q(k) = 4k^2. \tag{3}$$

Both polynomials can be implemented into an FPGA, the first one needs more calculation, whereas the second option offers reduction of the polynomial size, reducing computational cost. However, using a lookup table the size is not a problem.

(4) *Renormalization of the Ranges of the Fuzzy Variables.* Convert the normalized range to the previous range of the system.

The method can be applied to both inputs, as it was shown in [22, 23]; in the first work, two different tune factors were used, one for each fuzzy input. In the second work, experiments modifying both inputs using the same tuning value were shown. Results of both methods are satisfactory, using [23, 24] requires fewer resources and is easier to tune the system using just one variable.

6. Experiments and Results

With the aim of evaluating equivalent implementations of high-performance FLC embedded into an FPGA; two different systems were tested.

6.1. System 1: Speed Control of a DC Motor. For system 1, two sets of experiments were conducted. The first set comprises six experiments using different development platforms where the FLC was implemented to solve the previously regulation of speed problem. In the second set, the configuration of ARM soft processor with the FLC IP was chosen, because it demonstrated to be the best for the target of this work; therefore, we tested the controller using the STA to tune the controller.

For All Experiments. We consider a complete fuzzy inference, the process of fuzzification of crisp data, to infer a conclusion using the Mamdani inference engine, and defuzzification to obtain a crisp value.

Next, for each experiment, particular characteristics are described. Table 1 summarized the results.

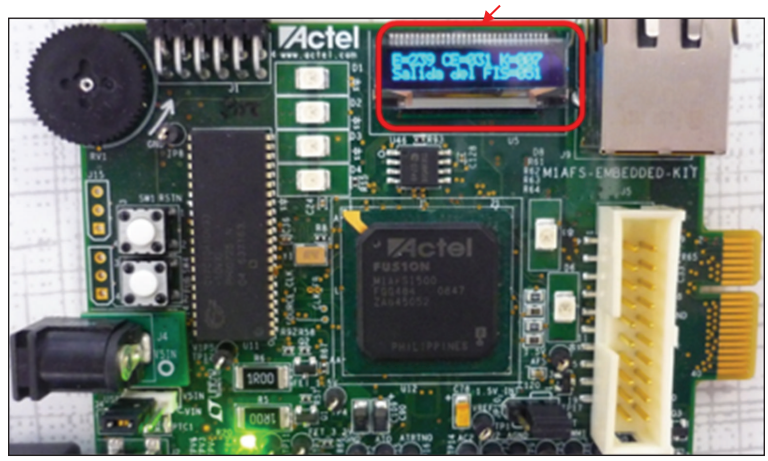


Figure 11: Actel development board for FPGA model "Embedded M1AFs1500 KIT."

Table 1: Comparing performance of FLC implementation for different development platforms.

Experiment	Platform	CPU	Clock frequency	Runtime (ms)
1	PC	Core 2 duo	2.66 GHz	20
2	Spartan 3	Microblaze	50 MHz	37
3	Virtex 5	Microblaze	100 MHz	16
4	Virtex 5	Power PC	100 MHz	12
5	Atmel AVR	8-bit Microcontroller	16 MHz	87
6	Fusion	ARM with fuzzy coprocessor	50 MHz	0.000160

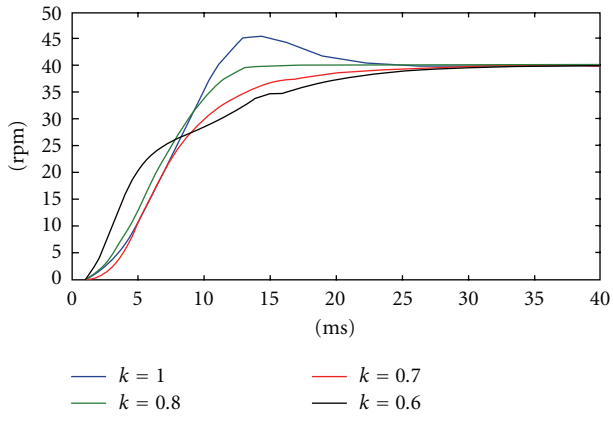


Figure 12: System response for different tuning factors.

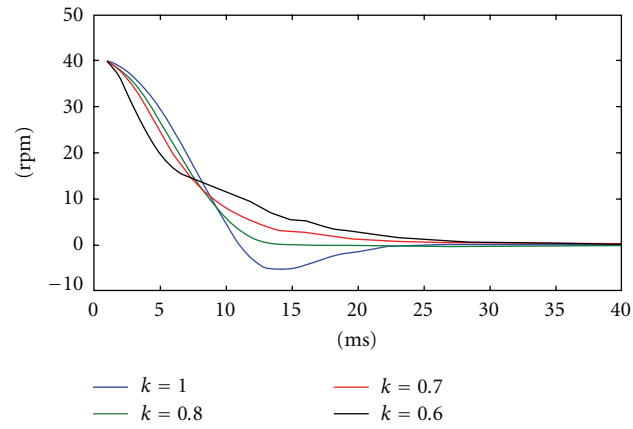


Figure 13: Plot of errors in the system response produced by different k tuning values. Note that $k = 0.8$ produces less errors.

6.1.1. Set 1 of Experiments

Experiment 1. The PC system is based on an Intel Core 2 Duo, with 6 GB of RAM. The FLC was implemented using the Fuzzy Logic Toolbox from Matlab-Simulink. The FLC writes and read information of the control plant using serial communication. A complete inference last 20 ms. We did not consider in the calculus the time due to serial communication.

Experiment 2. We used the Spartan 3 FPGA mounted in the Starter kit of Xilinx. We implemented into the FPGA the 32-bit Microblaze soft-core with 1 MB of RAM; the system clock is 50 Mhz. The FLC was implemented using C language. As the operating system we used the kernel of Xilinx standalone.

Experiment 3. In this test, a Virtex 5 FPGA mounted in the experimental board "ML507 FPGA technology" from Xilinx was used to embed a Microblaze soft-processor system, with 16 MB of RAM, and 2 KB of cache memory; the system clock is 100 Mhz. The FLC was implemented using

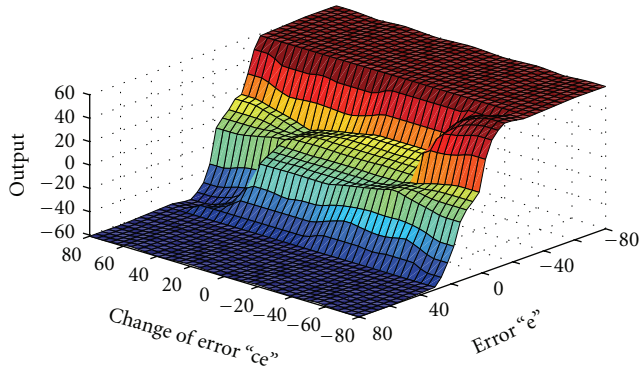


Figure 14: Surface control for a tuning factor $k = 0.5$.

ce \ e	LN	N	Z	P	LP
LN	LL	LL	LL	L	R
N	LL	L	L	Z	R
Z	LL	L	Z	R	RL
P	L	Z	R	R	RL
LP	LL	R	RL	RL	RL

Figure 15: Rule matrix for the inverted pendulum on a cart system.

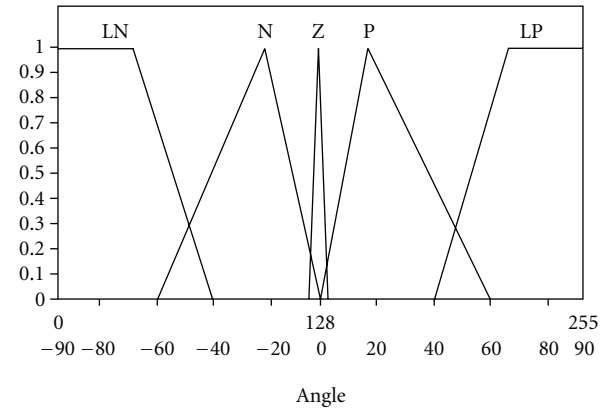
C language. As the operating system we used the kernel of Xilinx standalone.

Experiment 4. The Power PC hard processor of the FPGA Virtex 5 was used. The experimental board "ML507 FPGA" provided us with 16 MB of RAM and 2 KB of cache memory; and the system clock is 100 Mhz. The FLC was implemented in C language. As the operating system we used the kernel of Xilinx standalone.

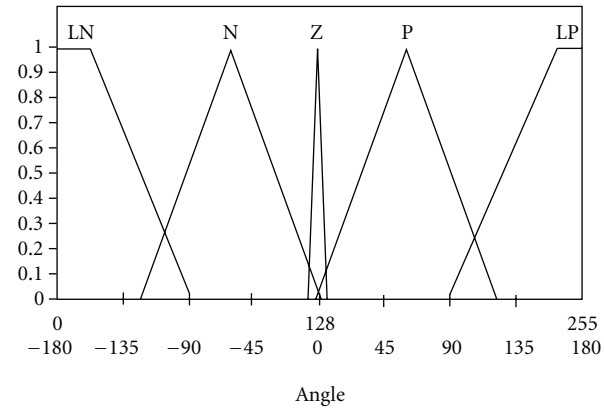
Experiment 5. The idea behind this experiment is to evaluate the FLC implemented in an 8-bit microcontroller system. We used the Atmel AVR development platform with the general-purpose microcontroller Atmel ATmega 16 running at a frequency of 16 Mhz.

Experiment 6. We used the system architecture of Figure 5 implemented in the Actel development board for FPGA model "Embedded M1AFS1500 KIT." This board is based on the FPGA Fusion of the same company; see Figure 11. This implementation uses the ARM cortex soft-processor and incorporates the FLC IP as a coprocessor. The FLC was coded in VHDL. Figure 11 shows a simple test of the FLC, we chose values with known output. The ARM soft-processor sent to the FLC, through the GPIO IP the input values "Error = 69.3 ($0 \times$ EF)", "change of error = −60.5 ($0 \times$ 1F)", and "$k = 0.5$ (0×07)". The output value of −60 (0×51) is read by the ARM and sent to the OLED display (red rectangle).

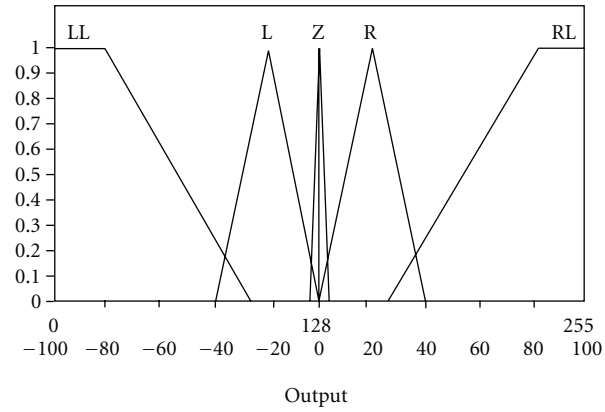
6.1.2. Set 2 of Experiments. Several experiments for different k values using the FPGA Fusion configured as in Figure 5 were achieved. The ARM soft processor handling the FLC IP



(a) Input error



(b) Change of error



(c) Output

Figure 16: The FLC has two inputs: Error and Change of error, and one output. The output provides a value of increment/decrement to be used by the integral part of the PD+I controller. The first scale [0.255] are the real MFs bit values definition in the FLC. The second scale $[-90, 90]$, $[-180, 180]$, and $[-100, 100]$ are the operation values for this application. The scale conversion is achieved by the ARM processor.

was tested. The system response was modified using the STA. The ARM processor modifies the k tuning factor though the GPIO IP.

Results for k values of 0.5, 0.7, 0.8, and 1.0 are shown. Figures 12 and 13 show the system response and errors, for the mentioned tuning factors. Note that $k = 0.8$ provides the

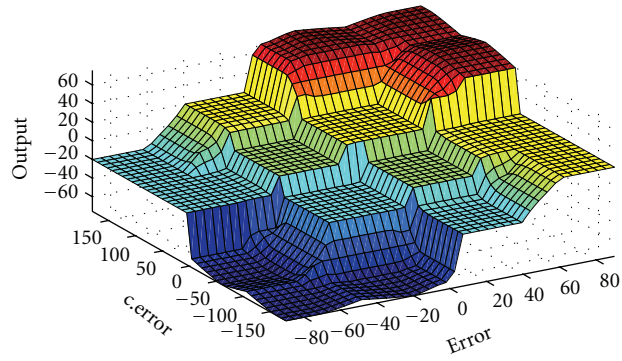


Figure 17: Surface control of the inverted pendulum system.

best system response; whereas $k = 0.5$ the slower response, and $k = 1.0$ the faster response with an overshoot. Figure 14 shows the control surface of the controller for a k value of 0.5. The sampling time for this set of experiments was 700 μs.

6.2. System 2: Inverted Pendulum on a Cart. For system 2 the inverted pendulum on a cart system of Figure 4 was used. We repeated all the experiments of Table 1 for this system; in congruence with the results of this table, we obtained practically the same running times in all the experiments. Next, we show the experimental setup for the FPGA Fusion. The membership functions were tuned using the STA.

The FLC has two inputs, "Error" and "Change of error" and one output labeled as "output". Figure 15 shows the rule matrix for this system, in Figures 16(a), 16(b) and 16(c) are the membership functions for the inputs and output linguistic variables, and Figure 17 shows the surface control of the inverted pendulum on a cart system.

7. Conclusions

The design and implementation of a High-Performance FLC that incorporates an efficient and practical method to tune the controller to the plant was explained. The tunable FLC IP was integrated into a SoC based on the FPGA Fusion jointly with an ARM soft processor and tested using two electromechanical systems.

In the first system, the regulation of the speed of a DC motor was the control objective, being the target of the paper to evaluate the performance of hardware implementations of fuzzy controllers used as IP Cores, two sets of comparative experiments of the system performance were achieved. The aim of the first set was to compare the FLC IP coprocessor against other development platforms, including a Core 2 Duo PC system, three implementations of the FLC programmed in C language running on the Microblaze soft-core processor mounted into the Spartan 3, and Virtex 5 FPGA systems. One C language implementation of the FLC running into the Atmel AVR 8-bit Microcontroller, and one implementation of the FLC coded using VHDL, and mounted as IP into the FPGA Fusion. We achieved an experimental evaluation of time computational complexity of the above systems.

The worst performance system was the FLC implemented using the Atmel AVR 8-bit Microcontroller, with this experiment we got the worst-case bound.

In the average bound, we found the three software prototypes of the FLC programmed in C language running on FPGA platforms, together with the PC development. Differences among FPGA systems are mainly due to systems clock; this is more evident on the Spartan 3 running with half of the frequency clock with respect to other FPGA systems. In the PC, the problem was that the FLC was running on Simulink controlling the plant using serial communication. Due to the implementation characteristic, this was not a real time system.

In the second system, the same set of tests were achieved and the execution times were consistent with the results obtained in the first system, some nonsignificant changes in values of FLC implemented in C occurred. However; for the case of study, the FLC IP Core, always last the same time, this is because the execution time is tight to the system clock, the FLC always last three clock cycles, and the GPIO IP last five clock cycles, therefore the FLC IP last eight clock cycles.

The FLC IP was designed to work in real time, when the FLC finish a whole inference cycle (fuzzification, inference, defuzzification) it enables the IRQ output request that can be sent to the interrupt controller for real time applications, or latched to be read by an input/output port to work in polling mode for applications that does not require real time.

The best-case SoC is the FLC IP combined with the ARM soft-processor, the speed-up with respect to the other development platform is 78, 125 times.

There are other interesting points to remark, for example, the flexibility that FPGA based system provides to implement SoC; this is important because it is possible to design independent low-cost low-power consumption and inexpensive systems that can mix digital and analogical signals. The use of specialized software, the growing availability of resources such as IP allow to simplify the development process of real time systems. The FLC IP soft-core has the advantage that it is not dependable on the bus system, and it provides to the user an easy handling capability without the need of the knowledge of the internal functionality; on the other hand, because it is not encrypted, the content can be read.

Finally, the incorporation of specialized debugging and testing modules into the SoC facilitates these complex and tedious tasks.

At present time, we are developing the IP core a Type-2 FLC, we have simulated stage by stage, for example the type-2 defuzzification stage [25]. The whole Type-2 FLC can be tuned using the STA algorithm for type-2 [26].

Acknowledgments

The authors would like to thank the "Instituto Politécnico Nacional (IPN)", "Comisión de Operación y Fomento de Actividades Académicas (COFAA)", and the Mexican "Consejo Nacional de Ciencia y Tecnología (CONACYT)" for supporting the research activities.

References

[1] M. J. Flynn and W. Luk, *Computer System Design: System-on-Chip*, John Wiley & Sons, Hoboken, NJ, USA, 2011.

[2] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, and K. Bouallaga, "Fully integrated FPGA-based controller for synchronous motor drive," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4006–4017, 2009.

[3] F. Sun, H. Wang, F. Fu, and X. Li, "Survey of FPGA low power design," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP '10)*, pp. 547–550, August 2010.

[4] D. Saha and S. Sur-Kolay, "SoC: a real platform for IP reuse, IP infringement, and IP protection," *VLSI Design*, vol. 2011, Article ID 731957, 10 pages, 2011.

[5] L. Tian, H. Pan, and D. Li, "Efficient Memory Processors Design of Multiple Applications for Multiprocessors Architecture," in *Software Engineering and Knowledge Engineering: Theory and Practice in Advances in Intelligent and Soft Computing*, Y. Wu, Ed., pp. 693–697, Springer, Berlin, Germany, 2012.

[6] R. E. Precup and H. Hellendoorn, "A survey on industrial applications of fuzzy control," *Computers in Industry*, vol. 62, no. 3, pp. 213–226, 2011.

[7] J. Kackprzyk, "Multistage fuzzy control: a model-based approach to fuzzy control and decision making," *Journal of Multi-Criteria Decision Analysis*, vol. 7, no. 4, pp. 239–240, 1998.

[8] T. J. Ross, *Fuzzy Logic With Engineering Applications*, John Wiley & Sons, Singapore, 3rd edition, 2010.

[9] L. Jim, *Embedded Control Systems in C/C++*, CMP Books, Berkley, Calif, USA, 2004.

[10] V. Thareja, M. Bolic, and V. Groza, "Design of a fuzzy logic coprocessor using handel-C," in *Proceedings of the 2nd IEEE International Workshop on Soft Computing Applications (SOFA '07)*, pp. 83–88, Oradea, Romania, August 2007.

[11] P. Sundararajan, *Performance Computing Using FPGAs*, Xilinx, San Jose, Calif, USA, 2010, http://china.origin.xilinx.com/support/documentation/white_papers/wp375.

[12] R. Sass and A. G. Schmidt, *Embedded Systems Design With Platform FPGAs*, Elsevier, New York, NY, USA, 2010.

[13] S. Anvar, O. Gachelin, P. Kestener, H. Le Provost, and I. Mandjavidze, "FPGA-based system-on-chip designs for real-time applications in particle physics," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 682–687, 2006.

[14] O. Montiel, J. Olivas, R. Sepúlveda, and O. Castillo, "Development of an embedded simple tuned fuzzy controller," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ '08)*, pp. 555–561, June 2008.

[15] O. Montiel, Y. Maldonado, R. Sepúlveda, and O. Castillo, "Simple tuned fuzzy controller embedded into an FPGA," in *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '08)*, pp. 1–6, May 2008.

[16] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Embedding a high speed interval type-2 fuzzy controller for a real plant into an FPGA," *Applied Soft Computing Journal*, vol. 12, no. 3, pp. 988–998, 2012.

[17] Y. Maldonado, O. Castillo, and P. Melin, "Optimization of membership functions for an incremental fuzzy PD control based on genetic algorithms," in *Soft Computing For Intelligent Control and Mobile Robotics*, O. Castillo, J. Kacprzyk, and W. Pedrycz, Eds., vol. 318, pp. 195–211, Springer, Berlin, Germany, 2011.

[18] N. R. Cázarez-Castro, L. T. Aguilar, and O. Castillo, "Fuzzy logic control with genetic membership function parameters optimization for the output regulation of a servomechanism with nonlinear backlash," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4368–4378, 2010.

[19] I. S. Shaw, *Fuzzy Control on Industrial Systems: Theory and Applications*, Kluwer Academic, New York, NY, USA, 2010.

[20] A. Stefano and C. Giaconia, "An FPGA-based adaptive fuzzy coprocessor," in *Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science*, C. Sandoval, J. Cabestany, A. Prieto, and F. Sandoval, Eds., vol. 3512, pp. 1–8, Springer, 2005.

[21] Actel, *The Advantages of the 32-Bit Cortex-M1 Processor in Actel FPGAs*, Actel, 2007, http://www.actel.com/documents/CortexM1_Advantages_WP.pdf.

[22] H. A. Ortiz-De-La-Vega, E. Gomez-Ramirez, and J. C. Cortes-Rios, "Simple Tuning Algorithm improvements for fuzzy logic controllers," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)*, pp. 1–8, July 2010.

[23] M. A. P. García, I. M. M. Sanchez, O. Montiel, R. Sepúlveda, and O. Castillo, "Simple tuning of a fuzzy pulse width modulation controller for a DC motor application," in *Proceedings of the International Conference on Artificial. Intelligence (ICAI '06)*, vol. 2, pp. 598–604, Las Vegas, Nev, USA, June 2006.

[24] O. Montiel, R. Sepúlveda, P. Melin, O. Castillo, M. Á. Porta, and I. M. Meza, "Performance of a simple tuned fuzzy controller and a PID controller on a DC motor," in *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, pp. 531–537, April 2007.

[25] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Modelling and simulation of the defuzzification stage of a type-2 fuzzy controller using VHDL code," *Control and Intelligent Systems*, vol. 39, no. 1, pp. 33–40, 2011.

[26] E. Gómez-Ramírez, P. Melin, and O. Castillo, "Simple tuning of type-2 fuzzy controllers," in *Soft Computing for Intelligent Control and Mobile Robotics, Studies in Computational Intelligence*, O. Castillo, J. Kacprzyk, and W. Pedrycz, Eds., vol. 318, pp. 103–123, Springer, Berlin, Germany, 2011.

Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2012, Article ID 419370, 8 pages
doi:10.1155/2012/419370

Research Article

A Novel Programmable CMOS Fuzzifiers Using Voltage-to-Current Converter Circuit

K. P. Abdulla and Mohammad Fazle Azeem

Department of Electronics and Communication Engineering, PA College of Engineering, Karnataka, Mangalore 574153, India

Correspondence should be addressed to K. P. Abdulla, kp.abdulla@gmail.com

Received 27 April 2012; Accepted 21 May 2012

Academic Editor: Oscar Castillo

This paper presents a new voltage-input, current-output programmable membership function generator circuit (MFC) using CMOS technology. It employs a voltage-to-current converter to provide the required current bias for the membership function circuit. The proposed MFC has several advantageous features. This MFC can be reconfigured to perform triangular, trapezoidal, S-shape, Z-Shape, and Gaussian membership forms. This membership function can be programmed in terms of its width, slope, and its center locations in its universe of discourses. The easily adjustable characteristics of the proposed circuit and its accuracy make it suitable for embedded system and industrial control applications. The proposed MFC is designed using the spice software, and simulation results are obtained.

1. Introduction

The concept of Fuzzy Logic was introduced by Zadeh nearly 40 years ago [1–6], and since then numerous applications of this theory has been implemented in various engineering and nonengineering fields. The numerous successful applications of fuzzy-control have sparked a flurry of activities in the analysis and design of fuzzy-control systems. Fuzzy-control theories have some salient features and distinguishing merits. The different fields in which fuzzy logic has been applied include signal processing, computer vision, automatic control, consumer electronics, household appliances, decision making analysis, and so on. The number of applications using fuzzy logic techniques to solve control problems has increased considerably. This rapid growth of fuzzy logic has motivated the researchers to go for efficient realization of fuzzy inference systems (FIS) [7, 8].

Fuzzy inference system can be implemented using software or hardware. Software implementation of FIS is useful when an application can be moduled to simulate, and calculate in advance, its multidimensional response characteristic. It provides flexibility as they usually support fuzzy systems with an arbitrary number of rules without any limitation concerning the number, type of membership, and range of inference mechanism. On the other hand, it works on a computer or a processor platform, and it has a drawback of being very slow [9, 10]. Hence it is not used for real-time application. A study of software and hardware implementation of FIS is described in [11], and it demonstrates the advantages of hardware design over software design in terms of speed and computational requirements. Therefore hardware realization is preferred for real-time applications, and various hardware techniques have been proposed [12].

The digital and analog techniques constitute the two existent approaches for the hardware realization of fuzzy systems. There are different techniques reported in the literature for the digital design of fuzzy systems. Among them, the most versatile methods are those which use field programmable gate arrays (FPGA) as reported by Maldonado et al. [13] and by Montiel et al. [14]. Even though the digital approach is superior in terms of ease of design, the analog fuzzy logic approach is proposed to get maximum efficiency in terms of silicon area, power consumption, and delay time or processing speed [15–19]. Thus it leads to low-cost and high-speed implementation compared to the digital counterpart. A direct interface of the controller to input and output continuous variables is also possible, thus making

the circuitry for analog-to-digital conversion unnecessary and inducing delay in the control loop too.

Fuzzification is an important concept used in the fuzzy logic theory. It is the process of converting a crisp value into a fuzzy value. The conversion is actually the incorporation of fuzziness, uncertainty, vagueness, or ambiguity in the crisp quantity. In the real world, the quantities which look crisp actually carry a considerable amount of uncertainty or vagueness. This uncertainty, vagueness, or imprecision in a fuzzy quantity can be represented by a membership function. Membership function is the key element in fuzzy signal processing. This provides a nonlinear relation that measures the compatibility of an object with the concept represented by a fuzzy set. Fuzzy set provides a convenient point of departure from the construction of a conceptual framework used in ordinary sets and have proved to have a much wider scope of applicability, particularly in the fields of pattern classification, information processing, statistical process control, and so forth. The researchers have developed various membership functions for the fuzzy processors using different techniques like digital methods [20, 21], mixed-mode signal [22, 23], and current-mode methods [24, 25]. The design of a membership function depends on the particular application to which the fuzzy processor is being applied.

The fuzzifier circuit resented in this paper is a suitable circuit to analyze and tune the nonlinear parameters of membership functions [26]. The important property of this fuzzifier is that by changing the various parameters, a membership function with different shape and slope can be easily generated. The fuzzifier designed here is a programmable circuit. This scheme is expected to find application in neuro-fuzzy processors where adaptation of the membership function parameters is required [27]. The result is a faster, cheaper, and higher controllable design compared to conventional fuzzification techniques. The proposed MFC has been laid out using full-custom techniques to provide standard cell. And it results in a CMOS-MFC standard cell, which could be then used as a standard component for implementing a number of required MFC in integrated circuit [28, 29]. This structure exploits the operation of the MOS differential amplifier as a current switch with soft transition region.

The major contributions of the proposed work can be summarized as follows: (1) firstly, the proposed design use a voltage-input, current-output interface which is very useful for the real world sensor interface with current-mode fuzzy-control systems, (2) secondly, the application of the analog voltage-to-current converter makes it easy to tune the nonlinear parameters of the MFC such as width and slope, and (3) finally, the low-power CMOS realization ensures a faster, cheaper, and highly controllable design for embedded system and industrial control applications.

This paper is organized into four sections: Section 2 describes the architecture of the proposed fuzzy membership function circuit and Section 3 explains the circuit level discretion for the same. It is followed by simulation results graphically described in Section 4, and conclusions are discussed in Section 5.

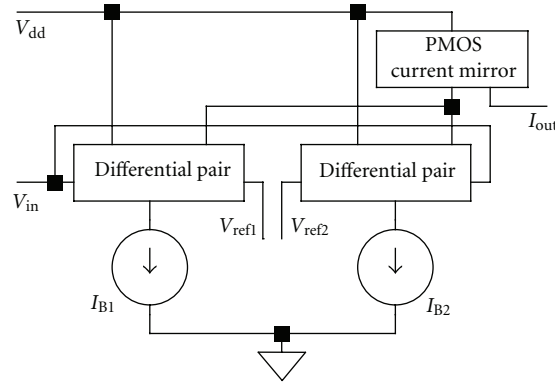


Figure 1: Block diagram of an MFC.

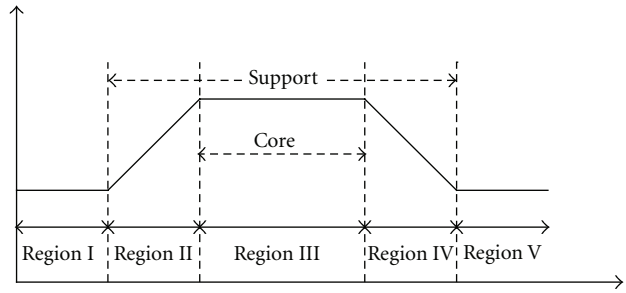


Figure 2: Trapezoidal shape of membership function.

2. The Proposed Circuit

Fuzzifying is the first step in a fuzzy system. The input data of a fuzzy logic controller are usually crisp values acquired from sensor measurement. Therefore, fuzzification interface is needed to calculate the belongingness (membership) of the observed inputs to the defined linguistic terms in the preconditions of the fuzzy rules. This is carried out through the fuzzy membership function circuits (MFCs), which performs a nonlinear transform from their inputs to their outputs. MFCs are provided with the input signal from the external environment, normally in the voltage mode. On the other hand, the processing steps following the fuzzification stage performs better in the current mode.The block diagram of an MFC is as shown in the Figure 1.

The MFC in Figure 1 is built of a coupled differential amplifier pair which has two differential pairs and a PMOS current mirror for replicating the drain currents to the output. The two reference voltages, V_{ref1} and V_{ref2}, where $V_{ref1} < V_{ref2}$, define the membership function. Depending on the relative values of input voltage V_{in} and the reference voltages V_{ref1} and V_{ref2}, the circuit operates in one of the five regions (I–V) as shown in Figure 2.

Basically, a fuzzy term can be defined by a membership function of trapezoidal shape as shown in Figure 2. The core of the function corresponds to the region which has full membership ($\mu = 1$), the ascending and descending boundaries are the regions with partial membership ($0 < \mu < 1$), and support is the sum of core and the boundaries in which $\mu > 0$. Further we assume that the current I_{B1} and I_{B2}

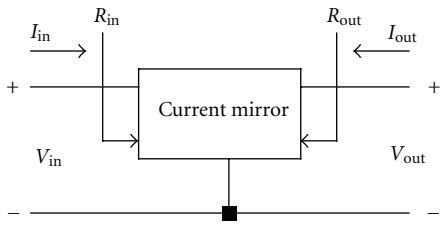


Figure 3: Block diagram of a current mirror.

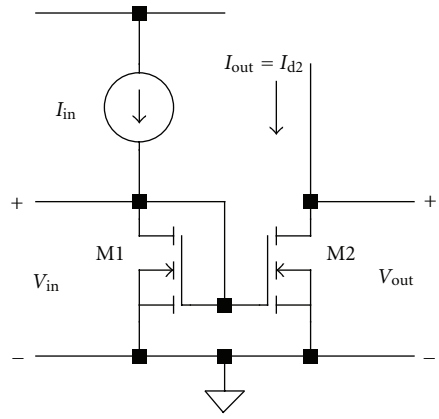


Figure 4: NMOS current mirror.

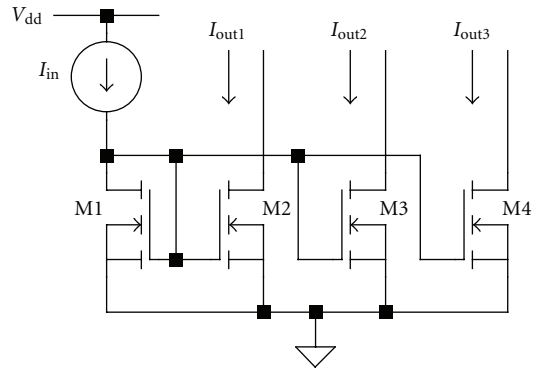


Figure 5: NMOS current replication circuit.

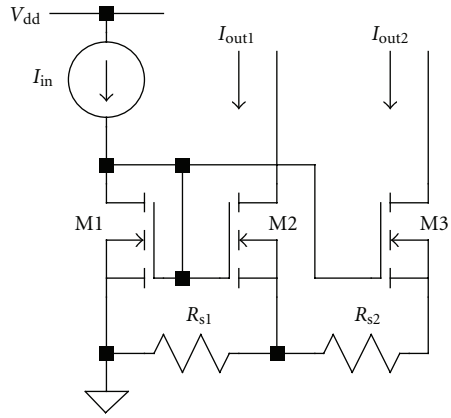


Figure 6: NMOS current mirror with two outputs and supply resistances.

in DP1 and DP2 are ideal and equivalent, the input devices in each differential pairs are symmetric, and the half widths of the transfer regions of DP1 and DP2 are V_1 and V_2, if W/L is equal for all MOSFETs used, therefore $V_1 = V_2 = \sqrt{2}V_{ov}$, where V_{ov} is the overdrive voltage.

The MFC shown in Figure 1 has four control signals—two voltage signals (V_{ref1} and V_{ref2}) and two current signals (I_{B1} and I_{B2})—for the width and slope tunability, respectively. In the proposed design of MFC, we have used an MOS voltage-to-current converter, which provides the bias current to the differential pairs DP1 and DP2. This voltage to current converter has a good linearity and very high sensitivity. The voltage-to-current converter was designed and simulated to study its characteristics and justify its usefulness in the design of tunable MFC circuits.

3. Circuit Level Description

As it can be seen from the Figure 1, the two important components of the proposed membership function circuit are the current mirrors and the differential amplifiers. This section explains the circuit level design and analysis of the basic analog structures used in the design of the proposed membership function circuit.

3.1. Current Mirrors. The block diagram of a typical current mirror is as shown in Figure 3. For an ideal condition, $I_{out} = (BI_{in})$, $R_{in} \gg 0$, and $R_{out} \gg \infty$, where R_{in} and R_{out} are the input and output impedances, respectively.

A current mirror is actually a combination of a diode-connected transistor followed by a single-transistor amplifier. The first one converts the input current into voltage, whereas the latter one converts the voltage into current. The current ratio will be quite accurate, simply because the non-linearity of the diode-connected MOS transistor (MOST) is compensated by the non-linearity of the amplifying MOST. If the ratio of their W/L's is B, then the current ratio is also B. Indeed, these MOSTs have the same V_{GS}, that is, ground-to-source voltage and hence $V_{GS} - V_T$. This ratio is the current gain.

3.1.1. NMOS Current Mirror. In the circuit shown in Figure 4, MOSFETs M1 and M2 form a current mirror and gate source voltages of both the MOSFETs are same because they are shorted. In MOSFET M1, the drain is shorted to its gate, thereby forcing it to operate in the saturation mode with

$$I_{in} = \frac{1}{2}K_n'\left(\frac{W}{L}\right)_1 (V_{GS1} - V_T)^2, \tag{1}$$

where channel length modulation is neglected. Now consider the MOSFET M2. It is also working in the saturation

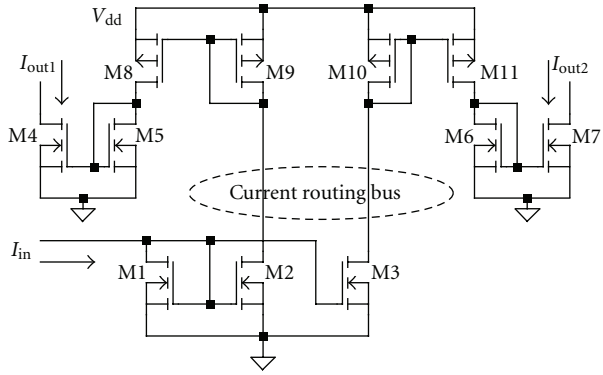


FIGURE 7: Bias distribution circuit using both current-routing and voltage-routing.

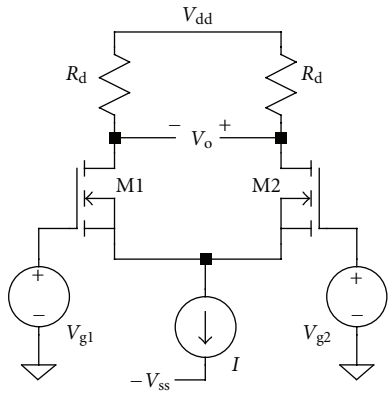


FIGURE 8: Differential pair with two inputs V_{g1} and V_{g2}.

mode, therefore the expression for its drain current (of the threshold voltages of M1 and M2 are equal) is given by

$$I_{out} = \frac{1}{2}K_n'\left(\frac{W}{L}\right)_2(V_{GS2} - V_T)^2. \quad (2)$$

Since the gates of M1 and M2 are shorted, $V_{GS1} = V_{GS2}$, therefore,

$$\frac{I_{in}}{I_{out}} = \frac{(W/L)_1}{(W/L)_2}. \quad (3)$$

Let $(W/L)_1 = (W/L)_2$, this implies that $I_{in} = I_{out}$.

Now if we would like to replicate more current of same value, then a current replication circuit is used by increasing the number of MOSFETs as shown in the Figure 5.

3.1.2. Current-Routing and Voltage-Routing. Consider the current mirror shown in Figure 6, which has one input and two outputs. At first, assume that $R_{s1} = R_{s2} = 0$. Also, assume that the input current is generated by a circuit with desirable properties.

Since the gate source voltage of M1 must be routed to M2 and M3 here, this case is referred to as an example of the *voltage routing* of bias signals. An advantage of this approach is that by routing only two nodes (the gate and the source

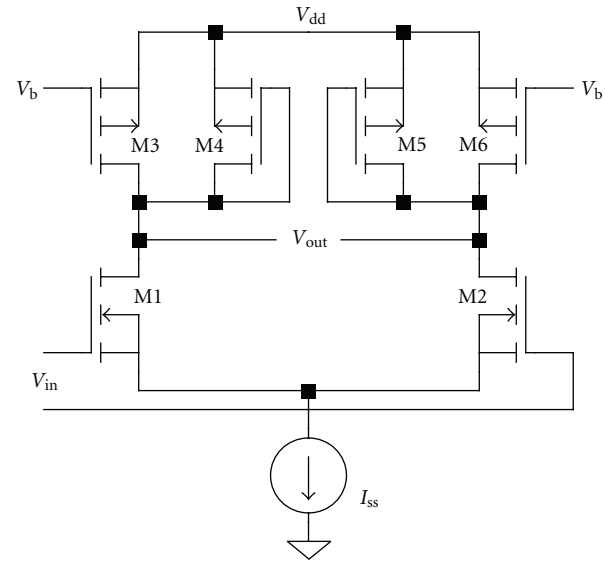


FIGURE 9: Differential pair with current source in addition to diode-connected load.

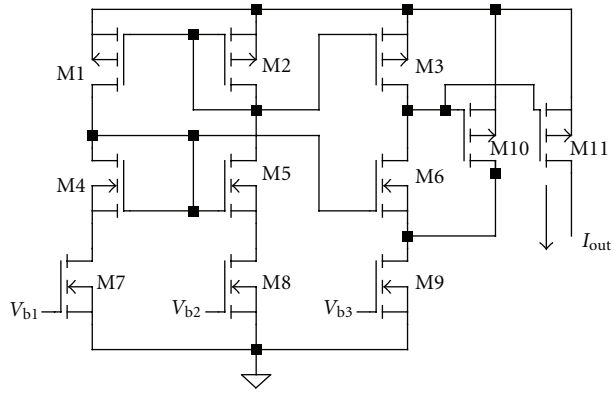


FIGURE 10: Voltage-to-current converter.

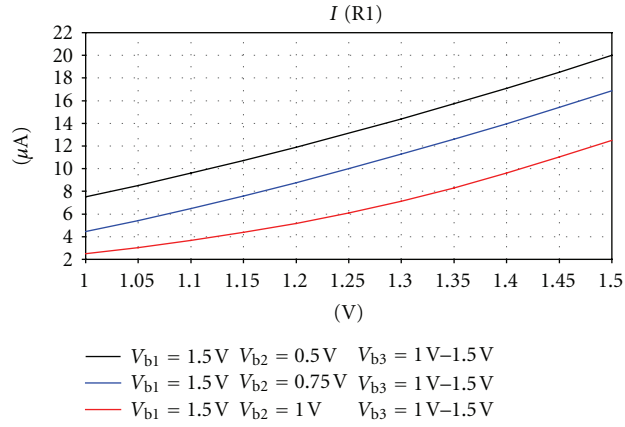


FIGURE 11: Voltage to Current converter output with $V_{b1} = 1.5$ V, $V_{b2} = 0.5$ V, 0.75 V, and 1 V, and V_{b3} sweep voltage from 1 V to 1.5 V in steps of 50 mV.

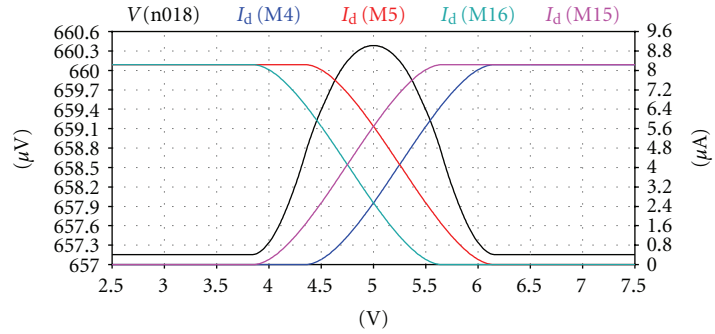


Figure 12: Gaussian-shaped membership function, $V_{b3} = 1.65$ V.

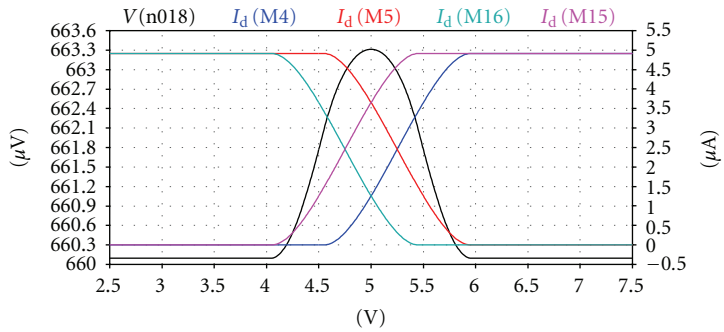


Figure 13: Gaussian-shaped membership function, $V_{b3} = 1.45$ V.

of M1) around the IC, any number of output currents can be produced.

Unfortunately, voltage routing has two disadvantages. First, the input and output transistors in the current mirror may be separated by distances that are large compared to the size of the IC, increasing the potential mismatches. In particular, the threshold voltage typically displays considerable gradient with distance across a wafer. Therefore, when the devices are physically separated by large distances, large current mismatch can result from biasing current sources sharing the same gate-source bias, especially when the overdrive is small. The second disadvantage of voltage routing is that the output currents are sensitive to variations in the supply resistances R_{s1} and R_{s2}.

To overcome these problems, the circuit in Figure 6 can be built so that M1–M3 are close together physically, and the current outputs I_{out1} and I_{out2} are routed as required on the IC. This case is referred to as an example of the current routing of bias signals as shown in Figure 7. Current routing reduces the problems with mismatch and supply resistance by reducing the distances between the input and output transistors in the current mirror compared to voltage routing. One disadvantage of current routing is that it requires one node to be routed for each bias signal. Therefore, when the number of bias outputs is large, the die area required for the "interconnect" to distribute the bias currents can be much larger than that required for voltage routing.

In practice, many ICs use a combination of current and voltage-routing techniques. If the current-routing bus in Figure 6 travels over a large distance, the parasitic capacitances on the drains of M2 and M3 may be large. In ICs using both current and voltage routing, currents are routed globally and voltages locally, where the difference between global and local routing depends on distance. When the distance is large enough to significantly worsen mismatch or supply resistance effects, the routing is global. Otherwise, it is local.An effective combination of these bias distribution techniques is to divide an IC into blocks, where bias currents are routed between blocks and bias voltages within the blocks. In this work, for the effective routing of the bias current, both current and voltage routings are used.

3.2. Differential Pair. MOS differential pair is widely used as an input stage in operational amplifiers and in many other types of circuits as well. This venerable circuit has a relatively large response to a change in the difference between its two input voltages, but a relatively small response to a change in the average value of its two input voltages. The usefulness of the differential pair stems from two key properties. First, cascades of differential pairs can be directly connected to one another without inter-stage-coupling capacitors. Second, the differential pair is primarily sensitive to the difference between two input voltages, allowing a high degree of rejection of signals common to both inputs.

Figure 8 shows an MOS differential pair. Here M1 and M2 are two matched transistors, biased by a common current

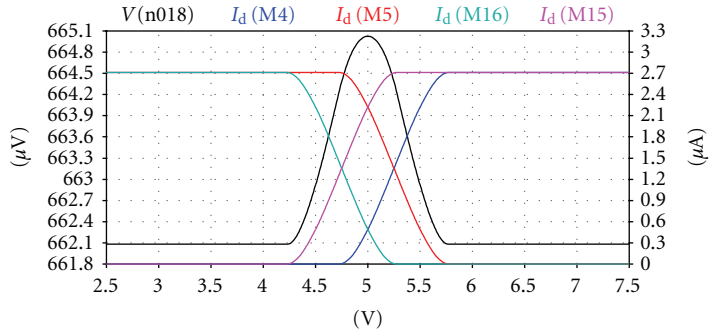


Figure 14: Gaussian-shaped membership function, $V_{b3} = 1.25$ V.

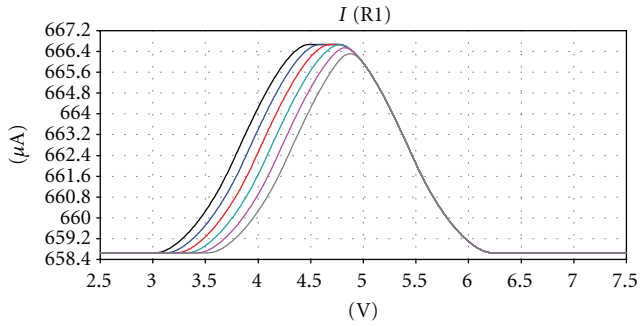


Figure 15: Width and position tunability through left side.

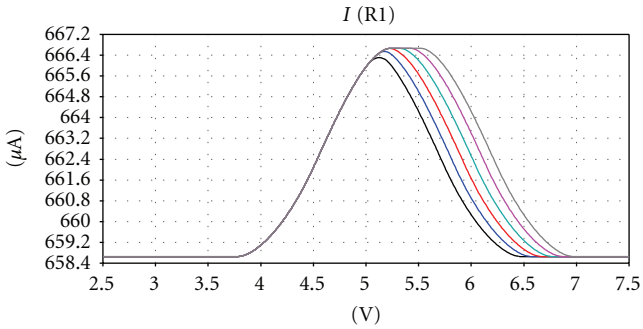


Figure 16: Width and position tunability through right side.

source $I_{applied}$, with identical drain resistance R_d. V_{g1} and V_{g2} are two input voltages applied at the gates G1 and G2.

The drain currents of the MOSFET are given by the following relation:

$$i_{d1} = \frac{I}{2} + \left(\frac{I}{V_{ov}}\right)\left(\frac{V_{id}}{2}\right)\sqrt{1 - \left(\frac{V_{id}/2}{V_{ov}}\right)^2},$$
$$i_{d1} = \frac{I}{2} - \left(\frac{I}{V_{ov}}\right)\left(\frac{V_{id}}{2}\right)\sqrt{1 - \left(\frac{V_{id}/2}{V_{ov}}\right)^2}. \quad (4)$$

When $V_{id} = \sqrt{2}V_{ov}$, $i_{d1} = I$ and $i_{d2} = 0$. When $V_{id} = -\sqrt{2}V_{ov}$, $i_{d2} = I$ and $i_{d1} = 0$.

3.2.1. Differential Pair with MOS Loads. The load of a differential pair need not be implemented by linear resistors. Differential pair can employ diode-connected or current-source loads. The diode-connected loads consume voltage headroom, thus creating a trade-off between output voltage swing, output voltage gain, and input common mode range. In order to alleviate the above difficulty, part of the current of the transistors can be provided by PMOS current sources. Here the idea is to lower the g_m of the load devices by reducing their current rather than their aspect ratio. For the proposed design, both the differential pairs shown in Figure 1 are connected with diode-connected and current-source loads to increase the gain as shown in Figure 9.

Figure 9 shows a differential pair with both diode-connected as well as current-source loads. This configuration can be used to increase the gain. For example, if M3 and M6 carry 80% of the drain current of M1 and M2, then the current through M4 and M5 is reduced by a factor of five. For a given $|V_{GSP} - V_{THP}|$, this translates to a factor of five reductions in the transconductance of M4 and M5, because the aspect ratio of the device can be lowered by the same factor. Thus, the differential gain is now approximately five times that of the case with no PMOS current source.

3.3. Voltage to Current Converter. As current-mode circuits are restricted to single fan-out, multiple current mirrors are required to share fan-out signals among several operational blocks. Voltage-mode inputs are thus preferable for fuzzy hardware systems, since they must be distributed to the membership function circuits of many rule blocks. Current-mode signals are appreciated afterwards, because of the advantages provided by current-mode processing. Tunable voltage-input current-mode membership function circuits are consequently useful building blocks to proceed fuzzification with current-mode analog hardware. For this reason, a voltage-to-current (V-to-I) converter is used to provide the bias currents in the proposed fuzzifier circuit. The schematic for the proposed voltage-to-current converter is as shown in the Figure 10.

In the above circuit, the MOS transistors M7, M8, and M9 are operated in the triode region, and a negative feedback loop is established by the MOS transistor M10. There the current through the MOS transistor M10 is B times up-scaled

current through the MOS transistor M11, which delivers the output current I_{out}, proportional to the input voltage ratio as given by the relation:

$$I_{out} = B\frac{V_{b3} - V_{b2}}{V_{b1} - V_{b2}}I_{in}, \quad (5)$$

where B is the ratio of W/L's of the MOS transistors M10 and M11, and effective I_{in} is the current through the MOS transistor M7. From the above expression, it is clear that the relationship between the input voltage V_{b3} and the output current I_{out} is linear and can be used as a voltage-to-current converter. In the proposed membership function circuit this voltage-to-current converter is used to tune the width and slope of the Gaussian-shaped membership function.

4. Simulation Results

All the above discussed circuits are designed and simulated using spice simulation software to study their characteristics and verify the working. This section provides the results of simulation for the above-discussed circuits and also a brief note on the obtained result.

The first result presented is the output for the proposed voltage-to-current converter as shown in Figure 11. Here we can observe that the V-I characteristics for the proposed circuit is linear and thus can be used as a voltage-controlled current source that can be used to supply the bias current for the membership function circuit. The required bias current can be set by varying the control voltage V_{b3} (V_{b1} and V_{b2} are fixed conveniently observing (5)).

Thus by varying the voltage V_{b3} we can control the width and the slope of the Gaussian-shaped membership function as shown in Figures 12, 13 and 14. For all the three simulation, $V_{b1} = 1.5\,V$ and $V_{b2} = 1\,V$.

Figures 15 and 16 shows the simulation results for the width and position tunability through left side and right side, respectively. For the left side tunability the voltage V_{low} is varied from 3.75 V to 4.25 V with 0.1 V increment. For right side tunability the voltage V_{high} is varied from 5.75 V to 6.25 V with 0.1 V increment.

From the above simulation results for the membership function circuit, it is clear that we have got three control voltages for the tuning of the membership function: V_{b3}, V_{low} and V_{high}. These voltages can be used to tune the slope, width and position of any particular membership function.

5. Conclusions

A new programmable CMOS fuzzifier circuit has been presented. The proposed membership function has the following advantages: proposed MFC is capable of generating different membership functions of different shapes—like Gaussian, S-shape, Z-shape, and trapezoidal—from a single circuit therefore it require lesser hardware. The proposed MFC exhibits a great programmability property just by use of three control voltages—V_{ref1}, V_{ref2}, and V_{b3}. Thus any characteristics of the MFC are easily obtained. The proposed circuit is very simple and compact and can be VLSI fabricated.

References

[1] L. A. Zadeh, "Fuzzy logic," *IEEE Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[2] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

[3] M. Sugeno, "An introductory survey of fuzzy control," *Information Sciences*, vol. 36, no. 1-2, pp. 59–83, 1985.

[4] E. H. Mamdani, "Applications of fuzzy algorithms for simple dynamic plant," *Proceedings of the Institution of Electrical Engineers*, vol. 121, no. 12, pp. 1585–1588, 1974.

[5] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller—part I," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404–418, 1990.

[6] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller—part II," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 419–435, 1990.

[7] T. J. Ross, *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, 2005.

[8] A. Costa, A. D. Gloria, F. Giudici, and Olivieri, "Fuzzy logic microcontroller," *IEEE Micro*, vol. 17, no. 1, pp. 66–74, 1997.

[9] H. Watanabe and D. Chen, "Evaluation of fuzzy instructions in a RISC processor," in *2nd IEEE International Conference on Fuzzy Systems*, pp. 521–526, April 1993.

[10] H. Watanabe, D. Chen, and S. Konuri, "Evaluation of min/max instructions for fuzzy information processing," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 369–374, 1996.

[11] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, "Embedding a KM type reducer for high speed fuzzy controller into an FPGA," *Applied Soft Computing*, vol. 12, no. 3, pp. 988–998, 2012.

[12] D. G. Zrilic, J. Ramirez-Angulo, and B. Yuan, "Hardware implementations of fuzzy membership functions, operations, and inference," *Computers and Electrical Engineering*, vol. 26, no. 1, pp. 85–105, 2000.

[13] Y. Maldonado, O. Montiel, R. Sepúlveda, and O. Castillo, "Design and simulation of the fuzzification stage through the Xilinx system generation," *Soft Computing for Hybrid Intelligent Systems*, vol. 154, pp. 297–305, 2008.

[14] O. Montiel, R. Sepúlveda, Y. Maldonado, and O. Castillo, "Design and simulation of the type-2 fuzzification stage: using active membership functions," *Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control*, vol. 257, pp. 273–293, 2009.

[15] M. Kachare, J. Ramírez-Angulo, R. G. Carvajal, and A. J. López-Martín, "New low-voltage fully programmable CMOS triangular/trapezoidal function generator circuit," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 10, pp. 2033–2042, 2005.

[16] T. Kettner, C. Heite, and K. Schumacher, "Analog CMOS realization of fuzzy logic membership functions," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 7, pp. 857–861, 1993.

[17] J. J. Chen, C. C. Chen, and H. W. Tsao, "Tunable membership function circuit for fuzzy control systems using CMOS technology," *Electronics Letters*, vol. 28, no. 22, pp. 2101–2103, 1992.

[18] N. Kong, U. Seng-Pan, and R. P. Martins, "A novel reconfigurable membership function circuit for analog fuzzy logic controller," in *Proceedings of the China Symposium on Circuits and Systems (CSCAS '07)*, 2007.

[19] L. Tigaeru, D. Alexa, and O. Ursaru, "New analog mode membership function circuit," in *Proceedings of the IEEE*

International Symposium on Signals, Circuits and Systems (SCS '03), pp. 601–604, 2003.

[20] D. Di Cataldo, V. Liberali, F. Maloberti, and G. Palumbo, "A/D conversion with fuzzy membership function," in *European Signal Processing Conference (EUSIPCO '96)*, pp. 1279–1282, Trieste, Italy, September 1996.

[21] F. A. Samman, R. S. Sadjad, and E. Y. Syansuddin, "The reconfigurable membership function circuit using analog bipolar electronics," in *Proceedings of the Asia-Pacific Circuits and Systems (APCCAS '02)*, pp. 537–540, 2002.

[22] N. Kong, U. Seng-Pan, and R. P. Martins, "A novel current-mode reconfigurable membership function circuit for mixed-signal fuzzy hardware," in *Proceedings of the 4th Regional Inter-University Postgraduate Electrical and Electronics Engineering Conference (RIUPEEEC '06)*, July 2006.

[23] L. Mesquita, G. Botura, and Saotome, "A mixed mode membership function circuit in CMOS technology," VI Workshop IBERCHIP, pp. 443-449, 2000.

[24] M. Conti, P. Crippa, S. Orcioni, and C. Turchetti, "Current-mode circuit for fuzzy partition membership functions," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 5, pp. V-391–V-394, June 1999.

[25] M. Sasaki, N. Ishikawa, F. Ueno, and T. Inoue, "Current-mode analog fuzzy hardware with voltage input interface and normalization locked loop," *IEICE Transactions on Fundamentals*, vol. 75, no. 6, pp. 650–654, 1992.

[26] D. D. Majumder, R. Bhattacharyya, and S. Mukherjee, "Methods of evaluation and extraction of membership functions—review with a new approach," in *Proceedings of the International Conference on Computing: Theory and Applications (ICCTA '07)*, pp. 277–281, March 2007.

[27] K. Basterretxea, J. M. Tarela, and I. del Campo, "Digital gaussian membership function circuit for neuro-fuzzy hardware," *Electronics Letters*, vol. 42, no. 1, pp. 44–45, 2006.

[28] A. Khodair, A. Badawy, and M. El-khatib, "VLSI Implementation of fuzzy logic membership function circuit," in *Proceedings of the 1st International Conference on Electrical Engineering*, March 1998.

[29] M. Tokmakçi, M. Alçi, and R. Kiliç, "A simple CMOS-based membership function circuit," *Analog Integrated Circuits and Signal Processing*, vol. 32, no. 1, pp. 83–88, 2002.