Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications

H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas

***Abstract*—The conventional digital hardware computational blocks with different structures are designed to compute the precise results of the assigned calculations. The main contribution of our proposed Bio-inspired Imprecise Computational blocks (BICs) is that they are designed to provide an applicable estimation of the result instead of its precise value at a lower cost. These novel structures are more efficient in terms of area, speed, and power consumption with respect to their precise rivals. Complete descriptions of sample BIC adder and multiplier structures as well as their error behaviors and synthesis results are introduced in this paper. It is then shown that these BIC structures can be exploited to efficiently implement a three-layer face recognition neural network and the hardware defuzzification block of a fuzzy processor.**

***Index Terms*—Bio-inspired, face recognition, fuzzy processor, imprecise computational blocks, neural networks, very-large-scale integration.**

I. Introduction

MANY of the existing scientific and engineering problems are solved using deterministic, precise, and explicit models and algorithms. These rigorous techniques that are known as hard-computing are normally used to solve a category of problems with similar properties such as high precision, predictability, and strict analysis [1]. These well-known and well-developed techniques are widely used to solve some of the existing engineering problems. However; these are in conflict with two frequent observations [2]. The first is that most of the real-world applications and problems are naturally imprecise and with low degrees of certainty while the second one is that using higher precision than the minimum necessary imposes higher implementation cost.

The soft-computing paradigm uses the above two guiding principles [2] as its fundamental base to enhance the traditional hard-computing problem-solving strategy. Soft-computing is an evolving collection of methodologies, which aim to exploit tolerance for imprecision, uncertainty, and partial truth to achieve robustness, tractability, and low cost [1]. Soft-computing methodologies are very popular today, because they have formed a major stream in practice of modern artificial intelligence. Neural networks (NN), fuzzy logic, and evolutionary computing are the core methodologies of soft-computing. Looking at the number of publications on various combinations of these three methods and their applications, it is concluded that the soft-computing methodologies have already been advantageous in numerous areas [1].

We introduce an important problem associated with conventional implementations of soft-computing systems in Section II. In Section III, we present our proposal to alleviate the problem and deliver more efficient implementations at improved cost and speed. Sections IV and V include the simulation and synthesis results to show how our solution can be applied to efficiently implement a three-layer NN for face recognition as well as a defuzzification block that is used in a fuzzy inference engine. The last section concludes the paper.

II. Problem Description

All soft-computing methodologies are based on the fact that most of the real-world applications and problems tolerate some natural imprecision and embody some degrees of uncertainty [2]. Therefore, it is not recommendable to develop and use deterministic, explicit, and precise models and algorithms to solve such problems. On the other hand, it is a fact that practical soft-computing applications are mostly implemented using digital hardware [3]–[12] that is based on binary logic with high degree of predictability and precision. A key problem arises here while a naturally precise and certain implementation bed should be used to realize the methodologies that are intrinsically relaxed to have higher imprecision and uncertainty. This conflict suppresses the main advantage of the soft-computing methodologies (i.e., their inherent tolerance to accommodate some imprecision and uncertainty) when they are implemented using conventional digital hardware.

The analog implementations of the soft-computing applications try to solve this problem by choosing a more relaxed implementation bed. An analog implementation is more efficient in terms of chip area, processing speed, and power consumption while this comes at the price of its limited accuracy with respect to its digital rivals [12]–[16]. However; digital technology has some unique features that prevent it to be replaced by analog implementations. Digital ICs are reliable, drift-free, and relatively noise immune, and have a repeatable and predictable behavior

Manuscript received December 28, 2008; revised May 02, 2009. First published December 31, 2009; current version published April 09, 2010.This paper was recommended by Asssociate Editor S. Callegari.

H. R. Mahdiani is with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14395-515, Iran, and also with the Computer and Electronics Department, Sh. Abbaspour University of Technology, Tehran 16589-53571, Iran (e-mail: mahdiany@ut.ac.ir).

A. Ahmadi, S. M. Fakhraie, and C. Lucas are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 14395-515, Iran (e-mail: a.ahmady@ece.ut.ac.ir; fakhraie@ut.ac.ir; Lucas@ut.ac.ir).

Digital Object Identifier 10.1109/TCSI.2009.2027626

[17]. Also the digital design process is simple and straightforward [11], [14] and there are many advanced digital system development tools that can be used to design complex digital systems with hundreds of millions of transistors. Furthermore, for analog implementation of a NN, there are several other issues to deal with [18] such as choosing a compatible training algorithm, nonvolatile weight storage and restoration, non-ideal response, and variations of the analog computational blocks [15], [18], [19]. The nanoelectronics, quantum, and molecular computing are the other most likely directions of the system development technologies [20]–[23]. However; it takes a long time to practically use them in commercial products. The next section introduces our proposed solution that helps to reconcile the discussed conflict without abandoning the digital realm.

III. The Solution: Using Bio-Inspired Imprecise Computational Blocks (BICs) Concept

Human brain as an intelligent computing entity has been a good inspiration source for development of some important artificial intelligence concepts such as NN and fuzzy computations. One of the important brain paradigms is that the collective behavior of the nature-made computational units in the brain when responding to computationally intensive tasks is such that it is less sensitive to the detail behavior of individual blocks. As a result, the total system continues to function properly despite possible imprecision or even failure in some computational units. This implies that from the implementation viewpoint, although the brain as a whole guaranties to preserve tremendous degrees of precision, performance, and reliability even in inconvenient situations, it is implemented using low precision neurons that perform their internal imprecise computations through transfer of ions across the cell membranes [24].

The inspiration that follows this paradigm is that it is not mandatory to use fully-precise and completely-deterministic building blocks for the implementation of bio-inspired systems. This inspiration from the nature-made computational blocks, relaxes the design process of the traditional human-made computational building blocks that might lead to different benefits such as decreased complexity and cost, or increased speed and performance. As an instance in digital VLSI territory, a designer might utilize some irregular adders and multipliers that are intentionally relaxed to provide an approximation of the computation result instead of its precise value. This relaxation also might imply decreased implementation costs of these bio-inspired computational blocks (BICs) in terms of area, delay, and power consumption with respect to the traditional precise components.

A traditional computational block (that provides the precise results of the computations) is entirely described with its physical properties or synthesis results such as area, delay, and power consumption. However; as a BIC structure provides approximations of the computations instead of their precise values, a complete description of a BIC should also include its error behavior (in addition to its synthesis results) that evaluates its result approximation approach. To clearly define the error behavior of a BIC structure, it is recommendable to represent or map its output results imprecision into some well known and well defined criteria. Some of the most useful error definition criteria that might be utilized to evaluate a BIC structure consist of: Probability of Error (PE) that indicates the probability of producing an erroneous result, Mean output Error (ME), mean relative error, Mean Squared output Error (MSE), MAXimum output error (MAX), and MINimum output error (MIN) [25]. All these factors are computable with comparison to the output results of a similar precise structure. This can be achieved performing some analytic studies or running some simulations.

Error behavior definition of a BIC is important because it guides us to properly design or pick suitable BIC structures for different application areas. For example, an imprecise adder structure which has a high PE is introduced in the next section. Although it mostly provides a different result with respect to a precise adder, it has a low output ME. The experimental results show that this adder might be efficiently used to implement a computationally intensive NN.

We introduce efficient instances of bio-inspired imprecise adders and multipliers. The error behavior and also synthesis results of the introduced imprecise adder and multiplier structures are discussed and compared with traditional precise ones. To provide the error simulation results of the individual BIC structures, the bit-true C models of the introduced blocks are prepared and added to our previously developed C Finite-Length Arithmetic Modeling Extension (C-FLAME) [26] bit-true modeling library. In addition, the error behavior of each BIC structure is presented using some analytic equations that are verified using C-FLAME simulation results. The synthesis results of the BIC structures are also presented by introducing an analytic study that is compared by practical synthesis results.

To show the applicability as well as the efficiency of the introduced BIC structures as a reliable approach for hardware implementation of soft-computing applications, we used them to successfully implement a three-layer face-recognition NN and the defuzzification block of a hardware fuzzy processing engine as two challenging cases. Instead of extensive further simulations, we suffice to the above two crucial applications and point out the following arguments. The genetic algorithm field does not involve any computations and is mainly based on random number generation. In fuzzy field, the fuzzy control is the most practical aspect of the fuzzy theory. The fuzzy control mostly consists of three distinct processes, fuzzification, fuzzy inference, and defuzzification. The defuzzification process in Mamdani fuzzy control is crucial among the others because of its high computational power demand with respect to fuzzification and fuzzy inference processes [5]. In NN field, the face-recognition benchmark discussed in the paper is a typical example of a computationally-intensive application and can be considered as a critical case analysis. Generally, if the computational load is not too demanding, then the need for hardware realization is not the first choice.

A. A Bio-Inspired Imprecise Adder: "Lower-Part-OR Adder"

1) "Lower-Part-OR Adder" Structure Description: Addition is a fundamental operation that can significantly influence the overall achievable performances [27]. The Lower-part-OR Adder (LOA) divides a p-bit addition into two m-bit and n-bit smaller parts $(m+n=p)$. As shown in Fig. 1, a p-bit LOA exploits a regular smaller precise adder (is called sub-adder) that

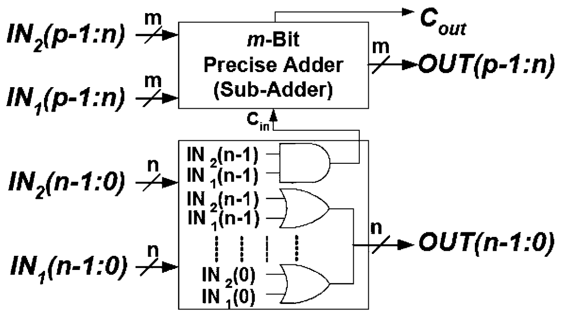


Fig. 1. Hardware structure of the Lower-part-OR Adder (LOA).

computes the precise values of the 'm' most significant bits of the result (upper-part) along with some OR gates that approximate the 'n' least significant result bits (lower-part) by applying bitwise OR to the respective input bits. An extra AND gate is used to generate a carry-in for the upper part sub-adder when the most significant bits of both lower-part inputs are one. This helps to take into account the trivial carries from lower to upper parts of the LOA adder to decrease its imprecision.

The sub-adder might have any desired precise structure such as carry look-ahead or ripple-carry adder, while using more efficient precise adders [27], [28] as the sub-adder directly leads to more efficient LOA. For a LOA with a constant Word-Length (WL or 'p'), higher 'n' or Lower-Part Length (LPL) values decrease the area, delay, and power consumption of the LOA while at the same time increase its imprecision. The error behavior and also the synthesis results of the LOA is presented and discussed in the following.

2) "Lower-Part-OR Adder" Error Behavior: The first parameter that might be useful to know about the LOA is that how often it provides an erroneous result (PE) as described in (1):

$$\mathrm{PE_{LOA}} = 1 - (3/4)^{\mathrm{LPL}} \tag{1}$$

Equation (1) shows that in a LOA, the error probability is independent of the WL and only depends on LPL. Fig. 2 shows that how fast the PE grows with increasing the LPL. It shows that for the LPL values of 2 and 8, the LOA provides about 40% and 90% erroneous results with respect to a precise adder of the same WL. Therefore, the LOA is not suitable to be used in applications that are sensitive to the number of errors regardless of their relative magnitude.

Although the LOA seems not to be a useful structure at the first glance based on its high number of erroneous results, it is important to discover more about the distribution and statistics of its error values. Equations (2) and (3) explain the maximum and minimum differences between the outputs of a LOA and a precise adder of the same WL. As shown by equations, the LOA provides both larger and also smaller results with respect to a precise adder based on its input values. Therefore, an important feature of the LOA is that it has a nearly unbiased and symmetric output error range between $[2^{\mathrm{LPL}-1} - 1, -2^{\mathrm{LPL}-1}]$ for signed or unsigned additions

$$\mathrm{MAX_{LOA}} = 2^{\mathrm{LPL}-1} - 1 \tag{2}$$
$$\mathrm{MIN_{LOA}} = -2^{\mathrm{LPL}-1}. \tag{3}$$

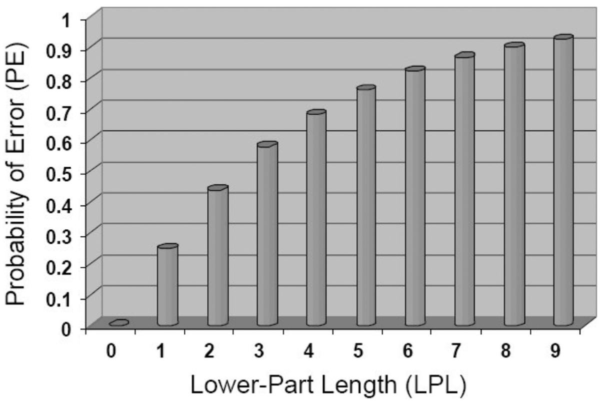


Fig. 2. Probability of producing an erroneous result by LOA for different LPL values.

Equation (4) demonstrates the mean output error of the LOA based on our analytic studies and simulation results. As another good feature, it shows that an LOA with any desired WL and LPL, provides a constant mean output error (that does not depend on the WL or LPL). This is achieved due to the symmetric output error range of the adder. Also the slightly negative bias of the ME value is due to the inequality that exists between MAX and MIN as shown in (2) and (3)

$$\mathrm{ME_{LOA}} = -0.25. \tag{4}$$

The mean squared error of the LOA is described using (5) for LPL > 0. It shows that the MSE grows exponentially with increasing of LPL

$$\mathrm{MSE_{LOA}} = 2^{2\times \mathrm{LPL}-4}. \tag{5}$$

3) "Lower-Part-OR Adder" Synthesis Results: Using the unit-gate model [29], [30], (6) and (7) describe the gate-count $(\text{Gate Count}_{\mathrm{LOA}}(\mathrm{WL}))$ and gate-delay $(\text{Gate Delay}_{\mathrm{LOA}}(\mathrm{WL}))$ complexities of LOA in general. The significance of these equations lies in their generality. They show that the gate-count and gate-delay of a WL-bit LOA depends on WL and LPL values, as well as the gate-count and gate-delay of its precise sub-adder with the length of *'WL-LPL'* (WL minus LPL) bits. They also show that in general, higher LPL values lead to smaller and faster LOAs with respect to the precise adder of the same WL

$$\begin{aligned}&\text{Gate Count}_{\mathrm{LOA}}(\mathrm{WL})\\ &\quad = \text{Gate Count}_{\text{Sub-adder}}\,(\mathrm{WL}-\mathrm{LPL}) + (\mathrm{LPL}+1)\end{aligned} \tag{6}$$
$$\begin{aligned}&\text{Gate Delay}_{\mathrm{LOA}}\,(\mathrm{WL})\\ &\quad = \text{Gate Delay}_{\text{Sub-adder}}(\mathrm{WL}-\mathrm{LPL}) + 1.\end{aligned} \tag{7}$$

The unit-gate model is chosen because it perfectly reflects the structure of the complex gates such as adders and multipliers by modeling their basic individual logic operators and so is commonly used in literature [30]. Comparison of the unit-gate model area/delay results with precise transistor-level synthesis results [31] show that this model yields acceptable relative accuracy at a high abstraction level using only simple analytic calculations [30]. To support these general analytic studies with a specific

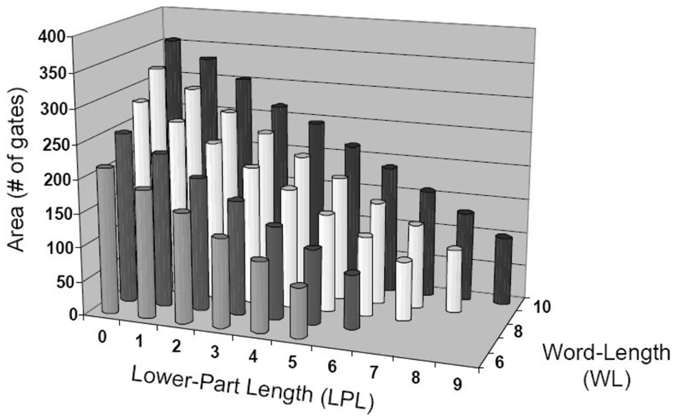


Fig. 3. LOA area for different WL and LPL values.

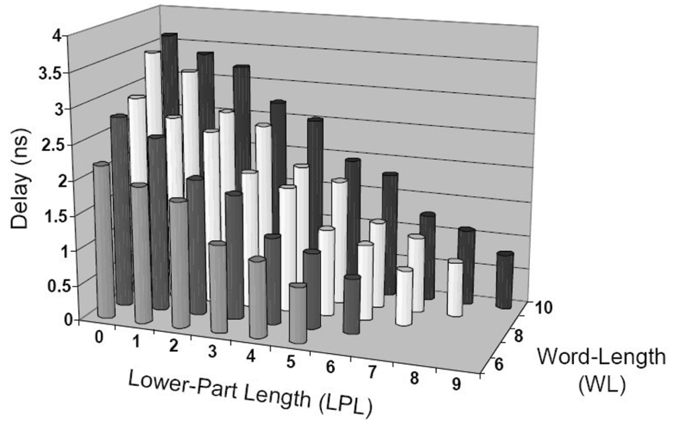


Fig. 4. LOA delay for different WL and LPL values.

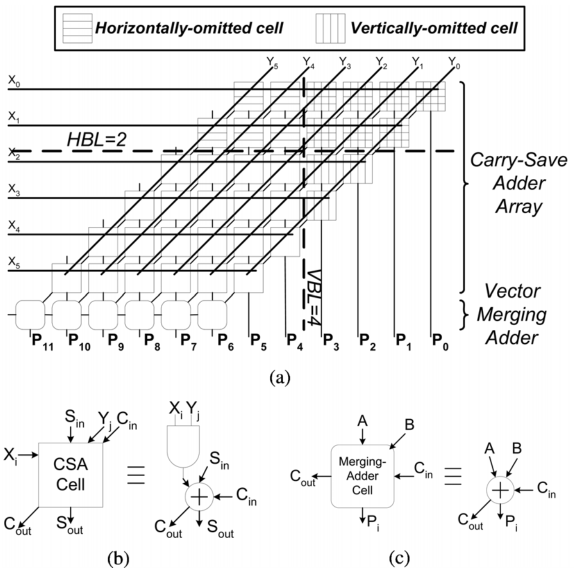


Fig. 5. (a) Hardware structure of the Broken-Array Multiplier (BAM), (b) CSA cell, and (c) vector merging adder cell.

real case study, Figs. 3 and 4 demonstrate our practical synthesis results of a LOA with ripple-carry sub-adder, for different WL and LPL parameter values when they are synthesized with 0.13 μCMOS technology library cells using Leonardo Spectrum synthesis tool. In both figures, a LOA with zero LPL value corresponds to a precise ripple carry adder of the same WL. Increasing LPL above zero reduces the length of the sub-adder that results in decreased area and delay of the whole LOA. Fig. 3 shows that for different WLs, the LOA gate-count decreases between 21 to 31 gates as LPL increases by 1 unit. Fig. 4 also demonstrates 0.16 to 0.55 ns reduction in LOA delay with 1 unit increment of the LPL parameter.

Considering LOA power consumption, a brief analytic study shows that it consumes less power with respect to a precise adder. The dynamic power consumption of a circuit which dominates its total power dissipation [32] is given by [33]

$$P_{\text{Dynamic}} = \alpha \cdot C_L \cdot V_{\text{DD}}^2 \cdot f \tag{8}$$

where α is the activity factor, C_L is the load capacitance, V_{DD} is the supply voltage, and f is the clock frequency. Therefore, as LOA has lower gate count as indicated before, it has lower total C_L with respect to a precise adder of the same WL. At the same time, as the LOA lower-part is constructed using lower activity OR gates, it also has smaller activity factor (α).

B. A Bio-Inspired Imprecise Multiplier Structure: "Broken-Array Multiplier" (BAM)

1) "Broken-Array Multiplier" Structure Description: Fig. 5(a) demonstrates the basic structure of a 6×6 broken-array multiplier (BAM) which is very similar to the structure of an array multiplier. An Array multiplier with a m-bit multiplicand and n-bit multiplier array multiplier consists of $m \times n$ similar cells called carry-save adder (CSA) array whose cells are shown by squares, followed by a m -bit vector merging adder (whose cells are shown by rounded squares) to convert the carry-save redundant form into regular binary format [34]. Each cell contains a 2-input AND gate that forms partial product and a full-adder (CSA) to add the partial product into the running sum [35]. Fig. 5(b) illustrates the internal gate-level structure of each CSA array and also vector merging adder cell [35]. The first row CSA cells in Fig. 5(a) however; might be replaced with only one AND or NAND gate. A BAM breaks the CSA array and omits some cells that lead to a smaller and faster circuit while provides approximate results. As shown in Fig. 5(a), the number and position of the omitted cells (that are hatched) depend on two introduced parameters: Horizontal Break Level (HBL) and Vertical Break Level (VBL). The $\text{HBL} = 0$ means that there is no horizontally omitted CSA cell. Increasing the HBL, moves the horizontal dashed line downward and all cells that fall above the dashed line are omitted. Similarly the $\text{VBL} = 0$ do not omit any CSA cells while as the VBL increases, the vertical dashed line moves left and all separated right-side cells are omitted. The respective output values of the all omitted cells are considered to be null. A similar technique might be applied to other multipliers such as Wallace-tree and Booth encoding structures to build new imprecise multipliers.

Although our main focus is on unsigned multiplication, the basic structure and operation of the signed and unsigned BAM are the same [35]. Their main difference is that in an unsigned BAM all the CSA cells are identical and can be omitted based

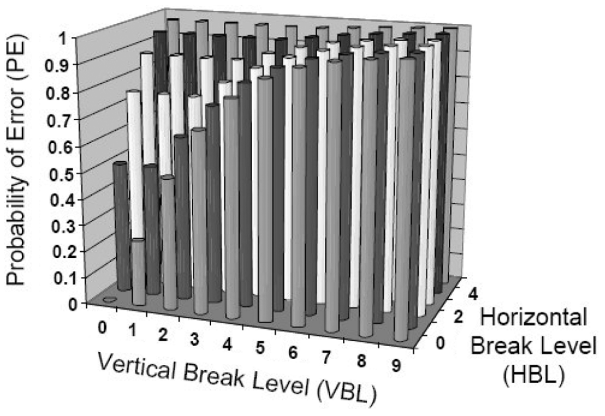


Fig. 6. Probability of producing erroneous result by BAM for different HBL and VBL values when $\mathrm{WL} = 10$.

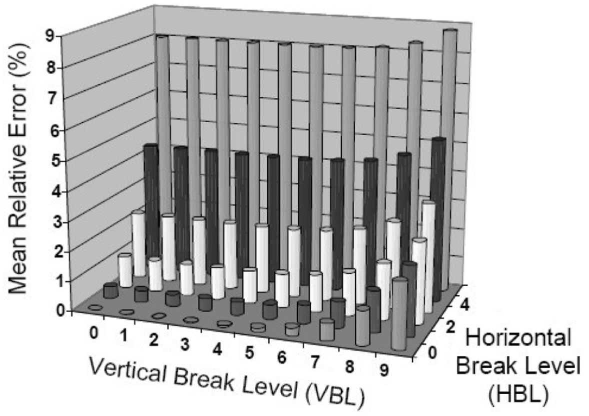


Fig. 7. BAM mean relative error for different HBL and VBL values when $\mathrm{WL} = 10$.

on HBL/VBL values while in a signed BAM, the CSA cells that operate on the sign bits of the partial products are not omitted at all to preserve the correct result sign. Therefore, increasing the HBL/VBL in an unsigned BAM leads to higher number of omitted CSA cells with respect to a signed BAM.

2) "Broken-Array Multiplier" Error Behavior: The accuracy of the BAM depends on three parameters: WL, HBL, and VBL. The simulation results show that increasing both HBL and VBL imply higher number of erroneous results while the HBL value has greater effects on the number of incorrect results than that of VBL. Fig. 6 demonstrates the probability of producing an erroneous result (PE) by BAM structure based on the values of HBL and VBL when $\mathrm{WL} = 10$. It shows that when HBL and VBL are both zero, $\mathrm{PE} = 0$ and the BAM provides 100% correct results and corresponds to a precise array multiplier completely.

The maximum and minimum differences between the output of a BAM and a precise multiplier of the same WL are determined by (9) and (10). They show that a BAM always provides smaller results with respect to a precise multiplier of the same WL

$$\mathrm{MAX_{BAM}} = (2^{\mathrm{WL}} - 1) \times \left(\sum_{i=0}^{\mathrm{HBL}-1} 2^i \right) + 2^{\mathrm{HBL}} \times \left(\sum_{i=0}^{\mathrm{VBL}-\mathrm{HBL}-1} (2^{\mathrm{VBL}-\mathrm{HBL}} - 2^i) \right) \quad (9)$$

$$\mathrm{MIN_{BAM}} = 0. \quad (10)$$

Equation (11) shows the mean error of the BAM in terms of WL, HBL, and VBL. To have a better understanding about the mean error values, Fig. 7 also demonstrates the mean relative error values of the BAM (in %) for $\mathrm{WL} = 10$ and different HBL/VBL parameter values. The relative error is computed by dividing the BAM output error by the precise result of the computation [25]. Fig. 7. shows that a BAM with $\mathrm{HBL} = \mathrm{VBL} = 0$ corresponds to a precise multiplier and provides a zero mean error. As the VBL and HBL increase, the relative ME of the BAM increases with different rates. For a constant HBL, increasing the VBL does not affect the BAM mean relative error and as a result mean error significantly. On the other hand, increasing the HBL itself by one unit, greatly affects the multiplier mean relative error. The figure shows that for the HBL values of 1 and 2, the output error of the BAM is less than 1% of the precise result. The simulation results show that HBL values of 1 or 2 are acceptable for most applications with different WLs

$$\mathrm{ME_{BAM}} = \left(\frac{2^{\mathrm{WL}} - 1}{4} \right) \times \left(\sum_{i=0}^{\mathrm{HBL}-1} 2^i \right) + 2^{\mathrm{VBL}-2} \times \left(\sum_{i=0}^{\mathrm{VBL}-\mathrm{HBL}-1} (1 - 2^{-(i+1)}) \right). \quad (11)$$

3) "Broken-Array Multiplier" Synthesis Results: Based on the analytic studies, (12) to (15) demonstrate the gate-count and gate-delay complexities of signed and unsigned BAM with 'm' and 'n' bit multiplicand and multiplier respectively. The equations are correct when HBL and VBL are less than 'm'. They show that the physical properties of a BAM depend on the size of the multiplicand and multiplier ('m' and 'n'), HBL and VBL values, and finally the gate-count and gate-delay of its 'm'-bit vector merging adder. It can be concluded that higher HBL/VBL values result in smaller and faster BAM structures. These equations also show that changing HBL has higher effects on area/delay of both signed and unsigned BAM with respect to VBL. As HBL has also greater effects on the error factors of BAM (as discussed before), the HBL and VBL should be treated as the coarse and fine tuning parameters of the area, delay, and precision of BAM respectively. Comparison of the gate-count and gate-delay equations of the signed and unsigned BAM also shows that for an unsigned BAM, increasing the HBL and VBL causes higher reductions in gate-count and delay with respect to its corresponding signed architecture

$$\begin{aligned}&\mathrm{GateCount_{UNSIGNED-BAM}}(m \times n)\\ &\quad = \mathrm{GateCount_{Merging-Adder}}(m) + m\\ &\qquad + \left[m \times (n-1) - m \times \mathrm{HBL} - \sum_{i=1}^{\mathrm{VBL}-\mathrm{HBL}} \mathrm{MIN}(i, m) \right] \times (\mathrm{GateCount_{CSA}} + 1)\end{aligned} \quad (12)$$

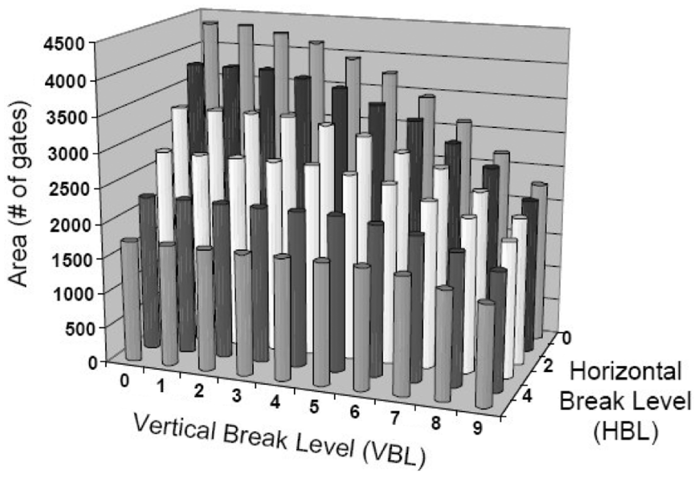


Fig. 8. BAM area for different HBL and VBL values when $\mathrm{WL} = 10$.

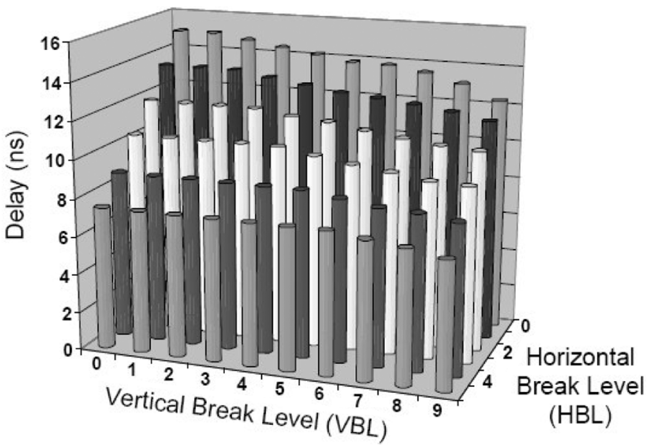


Fig. 9. BAM delay for different HBL and VBL values when $\mathrm{WL} = 10$.

$$\begin{aligned}&\mathrm{GateDelay}_{\mathrm{UNSIGNED-BAM}}(m \times n)\\&\quad = \mathrm{GateDelay}_{\mathrm{Merging-Adder}}(m) + 1\\&\qquad + (n - 1 - \mathrm{HBL}) \times \mathrm{GateDelay}_{\mathrm{CSA}}\end{aligned}\tag{13}$$

$$\begin{aligned}&\mathrm{GateCount}_{\mathrm{SIGNED-BAM}}(m \times n)\\&\quad = \mathrm{GateCount}_{\mathrm{Merging-Adder}}(m) + m\\&\qquad + \Bigg[m \times (n - \mathrm{HBL}) - (m - 1 - \mathrm{HBL})\\&\qquad - \sum_{i=1}^{\mathrm{VBL}-\mathrm{HBL}-1} \mathrm{MIN}(i, m)\Bigg] \times (\mathrm{GateCount}_{\mathrm{CSA}} + 1)\end{aligned}\tag{14}$$

$$\begin{aligned}&\mathrm{GateDelay}_{\mathrm{SIGNED-BAM}}(m \times n)\\&\quad = \mathrm{GateDelay}_{\mathrm{Merging-Adder}}(m) + 3\\&\qquad + (n - 1 - \mathrm{HBL}) \times \mathrm{GateDelay}_{\mathrm{CSA}}.\end{aligned}\tag{15}$$

To verify the discussed analytic studies about physical properties of BAM, Figs. 8 and 9 demonstrate the practical synthesis results of a 10×10 unsigned BAM with a ripple-carry vector merging adder for different HBL and VBL parameter values. The synthesis library is that of a $0.13\ \mu$CMOS technology. They verify that increasing HBL provides higher area and delay reduction than VBL. The figures show that the precise multiplier (that corresponds to a BAM with zero HBL and VBL) has the highest area and delay. As VBL and HBL grow above zero, BAM gains good area/delay savings with respect to the precise multiplier.

As to power consumption of the BAM, experimental results are not reported here. However; there is some previous work showing that preventing an array multiplier of always performing a full-length multiplication greatly reduces its power consumption [32], [36]. As a result, the BAM that always eliminates some part of the multiplication (based on the HBL/VBL values) is expected to yield some power reduction. Also as an analytic proof, as BAM has fewer number of gates with respect to a precise array multiplier, it has lower overall capacitance (C_L) and so based on (8) it has lower power consumption.

IV. Using BIC for VLSI Implementation of a Three-Layer Neural Network

This section contains the simulation and synthesis results to show the efficiency of using the introduced BIC structures for VLSI implementation of a computationally-intensive face-recognition three-layer NN.

A. BIC Implementation of a Neural Network for Face Recognition Application

The main objective of different implementations of a NN is to optimize three main constraints: accuracy, area, and processing speed [15] while preserving the scalability and flexibility of the network. In a digital VLSI implementation, the physical properties of a network tightly depend on cost, speed, and power consumption of the neurons. The most important factor that affects both the precision and physical properties of an individual neuron is its internal Word-Length (WL). Lower WLs lead to smaller and faster neurons. There are many reports on determining the efficient WL of different NN structures for different applications [15], [37]. The suitable WL of a NN depends on many parameters such as network structure, its number of layers and neurons, number of inputs, the application context, and its training method [15].

Our implemented NN is a three-layer network that is used for face-recognition application [38]. It consists of eight internal-layer neurons each with 960 inputs (and the same number of synaptic weights) that correspond to different pixels of a 32 by 30 grayscale photo. It also includes nine output neurons to determine different features of the input subject photo: the user's identification number (four neurons to encode 13 different persons), expression (two neurons to encode four different situations, sad, happy, angry, and neutral), user's head direction (two neurons to encode left, right, up, and straight), and if the user is wearing sunglasses or not (one neuron) [38].

The network is trained using back-propagation algorithm and with chip-in-the-loop approach [15]. The BIC model is prone for introducing new sources of imprecision. Therefore, using the chip-in-the-loop approach, the NN will also learn some of these imprecision sources and adjust the weights appropriately to account for them [18]. Also the network is considered to work properly only when it provides correct values (for all of its nine output neurons) for all of the testing and training patterns. The next two subsections compare the simulation and synthesis results of the network with BIC and with precise computational blocks to show that the BIC model of the NN is capable of tracking the behavior of a normal model while delivers lower implementation cost and higher performance.

B. Simulation Results

Two different C language bit-true models of the discussed NN are developed using C-FLAME. The first C model which is named precise or normal model consists of conventional adders and multipliers while the second C model is made using LOA and BAM structures and is called imprecise or BIC model. The Word-Length (WL) is the single controllable parameter of the normal model while the BIC model has three more parameters, namely LPL, HBL, and VBL. Choosing optimal value for WL in the precise model as well as four parameters of the BIC model (according to accuracy constraints of the application) [39], is a critical task that greatly affects performance, hardware cost, and power consumption [40]. It is possible to analytically determine the optimized parameter values of some simple systems however; when dealing with a complex nonlinear system, simulation is a more convenient and more feasible method [39], [41].

To have a fair comparison, two types of C simulation results are provided to observe and compare the external behavior as well as the internal activities of the two NN models. To observe the external behavior of the NNs, it suffices to determine if they are able to correctly recognize all features of all of the testing and training photos or not. However; it is more important to compare the internal activities of the two models to show that the BIC model can trace the internal behavior of the precise model correctly. To demonstrate the internal behavior of the networks, two different factors are measured, number of necessary training epochs (TE) to achieve the correct results and the network remaining mean squared error (MSE). The TE determines the convergence speed while the MSE stands for the convergence precision of the NN. The network MSE is determined as the average MSE values of the network outputs for all testing and training patterns based on [42]

$$\mathrm{MSE} = \frac{1}{P} \times \sum_{i=1}^{P} \left\{ \frac{1}{2} \times \sum_{j=1}^{N} (\mathrm{DesiredOutput}_j - \mathrm{AchievedOutput}_j)^2 \right\} \tag{16}$$

where 'N' and 'P' are the number of output neurons and input patterns respectively. The training phase is stopped if either the network MSE becomes less than a defined Epsilon (0.001) or the TE takes longer than a MAX_EPOCH (100). The first confirms the network has strongly detected all features of the input photos while the latter assumes that continuing the training does not improve the network MSE any more.

Fig. 10 shows the external as well as the internal behavior of the precise NN model for different WLs. The gray area in the diagram denotes the faulty external behavior of the network and specifies some areas in which the network provides at least one error. The network works properly in the white areas.

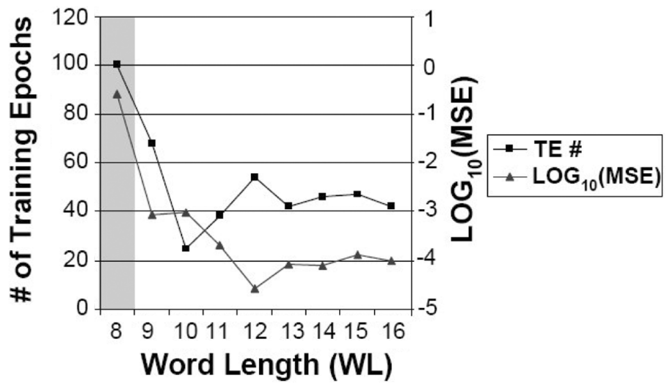


Fig. 10. Precise model behavior due to changes in WL.

According to Fig. 10, the $\mathrm{WL} = 9$ and $\mathrm{WL} = 13$ are two critical points of the simulation. For $\mathrm{WL} < 9$ the network fails to recognize at least one pattern. On the other side, increasing WL to more than 13 bits, does not improve any of the network internal activity indicators (MSE and TE#). Increasing WL from 9 to 13 does not improve the network MSE significantly and only causes the network to converge faster with fewer training epochs. Therefore, $\mathrm{WL} = 9$ leads to the smallest and fastest VLSI implementation of the network that correctly recognizes all of the input photos while its training convergence speed is not optimized. If the designer tends to achieve the best convergence speed, he should increase the WL to 13 and pay its extra cost and speed penalties.

While $\mathrm{WL} = 9$ leads to the most efficient normal implementation of the NN, the simulation results of the BIC model of the same WL show that it is able to track the external as well as the internal behaviors of the precise model of the same WL. Equations (6), (7), (14), and (15) show that choosing higher LPL, HBL, and VBL values decreases the cost and increases the speed of the BIC structures and the resulting NN model. However; increasing each of these parameters implies introducing higher imprecision in the corresponding BIC component and as a result adding higher errors inside neurons that might ultimately cause a neuron or the network to fail. Therefore, the maximum acceptable values of these parameters are limited and should be determined through simulations. Figs. 11–13 indicate the internal/external behavior of the BIC model when $\mathrm{WL} = 9$ with respect to changes in only one of its parameters (LPL, HBL, and VBL) while the two other parameters do not introduce any imprecision and are ineffective or zero. Again, the gray area in each diagram denotes the faulty external behavior of the network and specifies the regions that the network fails due to the intolerable amount of imprecision. The figures also imply that the LPL, HBL and VBL should be less than or equal to 4, 2, and 6 respectively. In such cases, the MSE and TE curves show that the BIC model tracks the precise model activities with negligible difference. It even sometimes provides better MSE values and less TE# due to the nonlinearity of the NN.

However, it should be mentioned that increasing the value of each parameter limits the maximum allowable values of the two others. It means that increasing the imprecision in one of the BIC

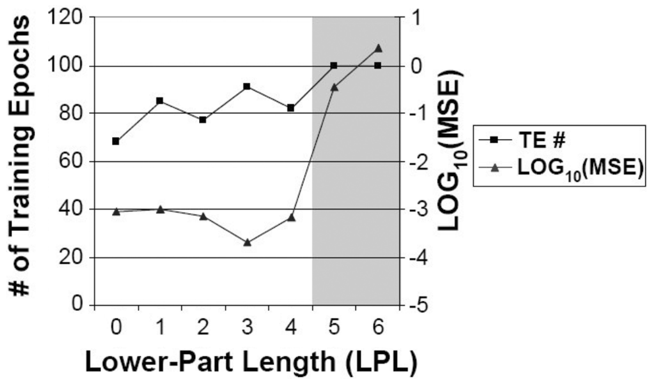


Fig. 11. BIC model behavior due to changes in LPL when $\mathrm{WL} = 9, \mathrm{HBL} = 0$, and $\mathrm{VBL} = 0$.

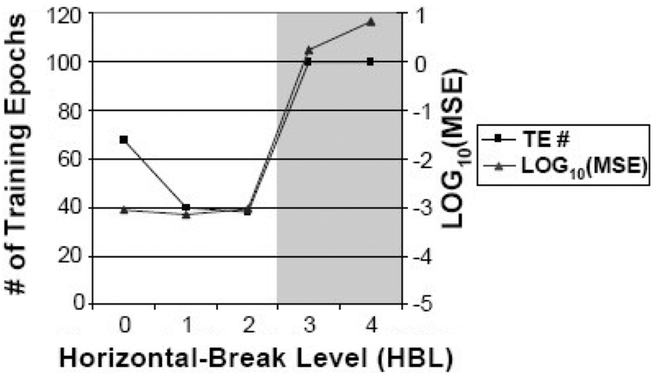


Fig. 12. BIC model behavior due to changes in HBL when $\mathrm{WL} = 9, \mathrm{LPL} = 0$, and $\mathrm{VBL} = 0$.

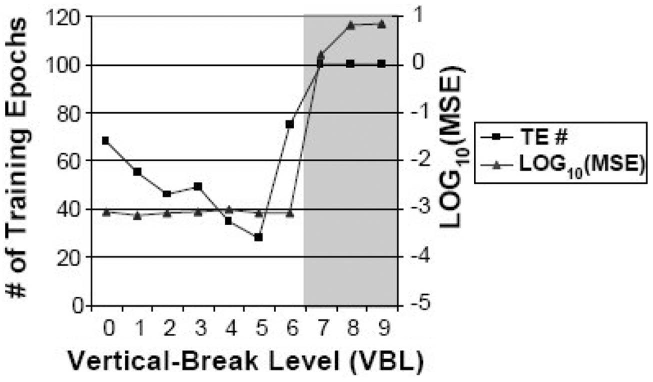


Fig. 13. BIC model behavior due to changes in VBL when $\mathrm{WL} = 9, \mathrm{LPL} = 0$, and $\mathrm{HBL} = 0$.

components inside the neuron, limits the amount of the tolerable imprecision in other BIC components. In other words, there is a correlation between the acceptable imprecision levels in different BIC components of a system that are determined by final values of these three parameters. Therefore, a final concurrent analysis is necessary to consider this correlation and simultaneously examine the allowable values of these parameters.

Table I includes the error simulation results of the precise and BIC models when $\mathrm{WL} = 9$. The results of the BIC model are presented for some of its best acceptable combinations of the LPL, HBL, and VBL parameters in which the BIC model works properly. The first column determines the type of the model

TABLE I
ERROR VALUES AND SYNTHESIS RESULTS OF THE PRECISE (P#1) AND BIC (B#1 TO B#3) MODELS WHEN $\mathrm{WL} = 9$

NN Model Type	Parameter Values				NN Simulation Results		NN Synthesis Results		
	WL	LPL	HBL	VBL	MSE	TE#	Delay* (ns)	Area** (# of gates)	Area × Delay
P#1	9	-	-	-	0.0008	68	17.98	4000	71920
B#1	9	2	2	6	0.0008	49	13.84	2586	35790
B#2	9	3	1	6	0.0006	69	14.96	2783	41633
B#3	9	4	0	5	0.0006	88	16.06	3179	51054

* Delay: critical path delay, ** Area: # of gates in data-path.

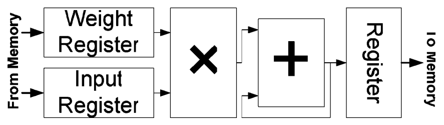


Fig. 14. Sequential architecture for VLSI implementation of a NN.

where 'P' stands for precise and 'B' for BIC models. The fifth and sixth table columns include the MSE and TE# simulation results of the precise and BIC models. It shows that B #1 and B#3 provide the best TE# and MSE among the others. The difference between MSE of the precise and anyone of the BIC models is in order of maximum 10^{-4} that is negligible. Although the BIC models provide similar MSE values, their main difference lies in their synthesis results which are discussed in the following section.

C. Synthesis Results

The face recognition NN might be implemented in several different ways such as fully-combinational, fully-sequential, or any other combination between these two extremes. In a fully-sequential implementation of a neural network, there exists only one hardware neuron that is scheduled to sequentially perform the computations of all neurons of the network one by one. In this paper, the fully-sequential structure that is shown in Fig. 14 is used to implement the NN both with normal and BIC components. Its data-path and critical-path consist of a single multiplier and an adder. The precise model is built using an array multiplier and a ripple-carry adder (RCA) while the BIC model exploits a BAM and a LOA. The sub-adder of the LOA, and the vector-merging adders of the multipliers in both models are RCAs.

The three last columns of Table I include the critical-path delay, data-path area without registers, and area × delay product for the precise and BIC models. The second row shows the synthesis results of the precise model. The following rows demonstrate the physical properties of the BIC model for some of its best parameter combinations as discussed. The table values show the synthesis results of the models on 0.13μCMOS technology library cells using Leonardo Spectrum synthesis tool.

Table values show that the normal implementation of the NN has respectively 12%, 20%, and 30% higher clock periods than BIC implementations while it also needs higher number of gates. Table values show that the precise model has 30%, 54%, and 101% higher delay, area, and area $\times$ delay product with respect to the BIC#1 while both of them recognize properly and have similar internal behaviors. As another instance, the precise model has about 20% longer critical path delay, 43% higher gate count, and 72% worse area $\times$ delay product with respect to the BIC#2 as another suitable parameter combination of the BIC model.

As a conclusion, the BIC implementation of the NN lead to some hardware with lower costs and higher speeds with respect to a normal implementation while their external behavior are entirely the same and there are only some differences in their internal activities and their number of training epochs.

The complementary simulation and synthesis results show that if we choose a WL more than 9, the area, delay, and power efficiency of the BIC over precise model improves more. Also choosing other implementation methods (such as combinational) or other adder/multiplier structures to implement the NN does not affect the natural efficiency of the BIC over normal implementation. However; they might lead to different combinations of the BIC model parameters due to their different synthesis results.

V. Using BIC for VLSI Implementation of the Defuzzification Block of a Fuzzy Processor

In this section, we show that how the introduced BIC sample structures might be exploited to enhance the realization cost and speed in VLSI implementation of the defuzzification block [5] as one of the critical issues in almost any fuzzy set theory application.

A. *BIC Implementation of the Defuzzification Block*

The main operation of the defuzzification process in a fuzzy system is to provide the linguistic to numeric conversion [43] that makes it possible to bind the fuzzy outputs of the system to common sensors and actuators. The most popular defuzzification method is the true Center Of Gravity (COG) [43]. It is the preferred method for most applications because of its unique features such as high precision and continuous and smooth change of its crisp output due to the continuous change of the controller inputs [44]. However, the most important drawback of this method is its high computational power demand and as a result implementation cost [45]. Therefore, there are a wide range of other defuzzification methods with less computational power demands and also less output accuracies [43]. Most of these new methods should be viewed as the simplified COG methods. In other words, the new methods are proposed to trade a piece of precision with reduced implementation cost and complexity [5] while they remain yet applicable. From the fuzzy control perspective, a defuzzification method is applicable whenever it can be used to solve a practical problem [43]. We observe that this cost-precision trade is similar to the same idea that is used in BIC. The main difference between these two concepts is that different defuzzification methods try to enhance the solution by reducing the precision at algorithmic level while the BIC reduces the precision at implementation level. Therefore, it is convenient to use BIC for the hardware implementation of different defuzzification methods as demonstrated below.

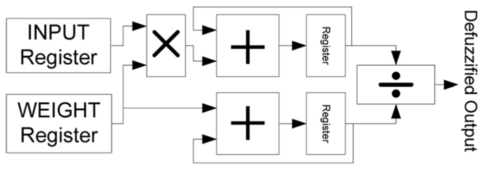


Fig. 15. Basic weighted-average structure for implementing different defuzzification methods.

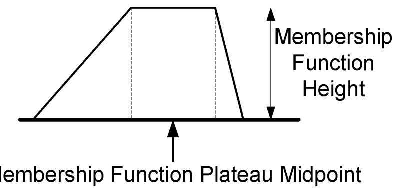

Membership Function Height
Membership Function Plateau Midpoint

Fig. 16. "Plateau Midpoint" and "Height" of a membership function which are used to compute the WPA defuzzified output.

Most of the existing defuzzification methods such as COG, Center Of Sum, Plateau Average, Weighted Plateau Average (WPA), Trapezoid Median Average (TMA), Trapezoidal Weighted TMA, and Rectangular Weighted TMA use a similar approach based on (17) to compute a weighted average of the fuzzy number membership functions as the defuzzified output [5]

$$\mathrm{DefuzzifiedOutput} = \frac{\sum_{i=1}^{N} \mathrm{Input}_i \times \mathrm{Weight}_i}{\sum_{i=1}^{N} \mathrm{Weight}_i}. \quad (17)$$

In the above equation, 'N' is the number of output membership functions, and Input_i and Weight_i are some specifications of the ith membership function which are determined by each defuzzification method. According to (17) the major difference between these defuzzification methods lies in their input and weight values. Fig. 15 illustrates a common hardware structure of all these methods which can be used to implement (17).

To provide an accuracy analysis of the defuzzification block using normal or BIC arithmetic blocks, it is necessary to focus on one of the above defuzzification methods. The WPA is picked because it is widely used in existing hardware fuzzy processors [5] due to its simple hardware implementation. In this method, the final defuzzified output value is computed as the weighted average of the midpoints of all fuzzy output plateaus based on [5]

$$\mathrm{WPA} = \frac{\sum_{i=1}^{N} \mathrm{Midpoint}_i \times \mathrm{Height}_i}{\sum_{i=1}^{N} \mathrm{Height}_i} \quad (18)$$

where 'N' is the number of output membership functions. Conforming (18) to the basic infrastructure of Fig. 15 and to (17) shows that the input and the weight in WPA are the plateau midpoint $(\mathrm{Midpoint}_i)$ and the height (Height_i) of the ith membership function respectively. Fig. 16 demonstrates that how the Midpoint and Height of a general membership function shape is determined.

B. *Simulation Results*

Two different C language bit-true models of the WPA are developed using bit-true C-FLAME library facilities and compo-

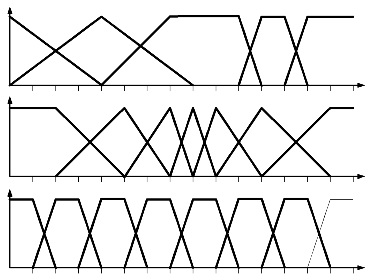


Fig. 17. Selected output membership function test cases.

nents. The first model which is named precise or normal model consists of some conventional precise adders and multipliers while the second model is made using LOA and BAM structures and is called imprecise or BIC model. The Word-Length (WL) is the single adjustable parameter of the normal model while the BIC model has three other parameters, that is LPL, HBL, and VBL. The target of this section is to show that the BIC model is able to track the behavior of the normal model with negligible (and also controllable) difference while has lower cost and higher performance. To have a fair comparison, three different error criteria are measured and compared for both models: Mean Absolute Error (MAE), Absolute Error Variance (AEV) and MAXimum absolute error (MAX) that are computed based on

$$\mathrm{MAE} = 100 \times \frac{1}{\mathrm{FuzzyWidth}} \times \left(\frac{1}{N} \times \sum_{i=1}^{N} |\mathrm{TrueCOG}_i - \mathrm{WPA}_i|\right) \quad (19)$$

$$\mathrm{AEV} = 100 \times \frac{1}{\mathrm{FuzzyWidth}} \times \left(\frac{1}{N} \times \sum_{i=1}^{N} (\mathrm{WPA}_i - \mathrm{MAE})^2\right) \quad (20)$$

$$\mathrm{MAX} = 100 \times \frac{1}{\mathrm{FuzzyWidth}} \times \{\forall_i\ \mathrm{Max}(|\mathrm{TrueCOG}_i - \mathrm{WPA}_i|)\}. \quad (21)$$

In the above, 'N' is the total number of fuzzy numbers, $\mathrm{TrueCOG}_i$ is the accurate center of gravity of the ith fuzzy number (that is computed using the true COG method as the base of comparison for error measurements) and the WPA_i represents the center of gravity estimation that is provided by the bit-true precise or imprecise WPA model that is under simulation. To more elaborate on the achieved results, the MAE, AEV, and MAX factors show the mean, variance, and maximum absolute change of the center of gravity with respect to the whole fuzzy number width (*FuzzyWidth*) in percentage.

To generalize the results, three different output fuzzy membership function sets with different properties and from different application areas [46] are selected as the test case on which the simulations are run. Fig. 17 illustrates the shapes of these membership function sets. As shown in the figure, these sets cover regular and irregular as well as triangular and trapezoidal membership functions to achieve more general simulation results.

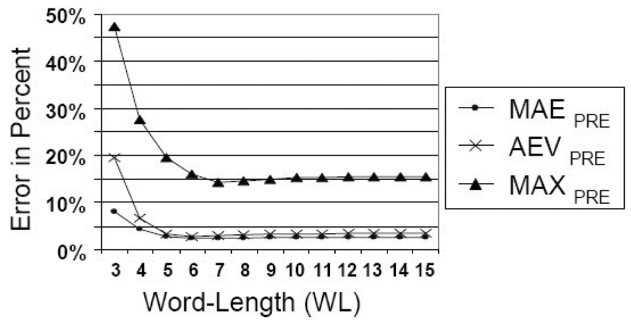


Fig. 18. MAE, AEV and MAX of the precise model with different WLs.

Fig. 18 shows the MAE, AEV and MAX curves of the precise bit-true model (PRE) with different WLs. The horizontal and vertical axes demonstrate the WL and error percentage respectively. There are two important notes about the curves. The first note is that increasing the WL more than six bits, does not affect any of the error factors significantly. So a WL of at least six is suitable for hardware implementation of WPA. The second observation that points to an important fact is that in contrast to the expectation that increasing the WL results in lower errors, all three error factors increase as the WL increases between 6 and 10 and then saturate to a constant value for WL $>$ 10. This occurs because the defuzzification output is formed by the division of two accumulation results as shown in (18). Although the error of the individual aggregations decreases and each saturates at a constant limit by increasing the WL, difference in the saturation rates for the numerator and denominator accumulations causes the discussed undesired fluctuations. This phenomenon might cause more significant fluctuations in case of BIC implementation due to their inherently added imprecision. So the defuzzification should be considered as a critical case that reveals the efficiency or weaknesses of using BIC structures.

Organizing the simulations for the BIC model is not as straightforward as for the precise model due to its higher number of parameters. Different approaches might be exploited to explore the BIC model parameter space and extract appropriate values for its four available parameters. In the previous section, we used a concurrent approach that simultaneously determines all the parameter values of the BIC structures which are used to implement the NN. In this section we introduce and use an iterative approach that determines and fixes the parameters one by one based on their effect on system error and/or physical properties. According to this approach, parameters that have greater effects should be fixed earlier (the coarse tuning) while the less effective parameters are used for further fine tuning. In defuzzification problem, the simulation results show the suitable order for fixing the parameters of the BIC model as first WL, then LPL, HBL and VBL.

Although choosing higher LPL, HBL and VBL values lead to faster and smaller BIC components and as a result defuzzification block, the maximum values of these parameters are limited by the application. As our discussion in this paper is to generally show the efficiency of the BIC units regardless of the application context, the parameter values are chosen in a manner to highly preserve the same error levels as those of the precise WPA with less than 1% relaxation that is suitable for most applications.

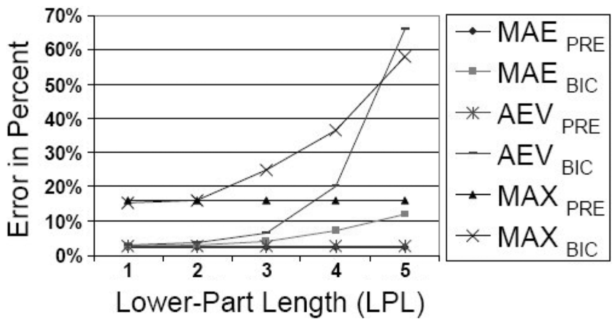


Fig. 19. MAE, AEV and MAX of the precise and BIC models for different LPLs while $\mathrm{WL} = 6, \mathrm{HBL} = 0$ and $\mathrm{VBL} = 0$.

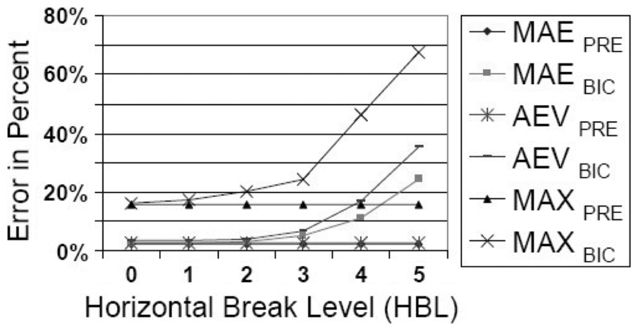


Fig. 20. MAE, AEV and MAX of the precise and BIC models for different HBLs while $\mathrm{WL} = 6, \mathrm{LPL} = 2$ and $\mathrm{VBL} = 0$.

To determine the WL while all other parameters are inactive (zero values for LPL, HBL and VBL that imply using precise adders and multipliers), the same results of the precise simulations are achieved (Fig. 18) that is $\mathrm{WL} >= 6$.

The LPL is the next parameter that should be tuned with inactive HBL and VBL values (i.e., inserting only the imprecise adder into design first). The results show that for any desired WL, the LPL values of '*WL-4*' and '*WL-5*' lead to less than 1% and 0.3% worse mean and variance errors with respect to the precise WPA model of the same WL. For example when $\mathrm{WL} = 6$ (Fig. 19), the BIC model provides 0.9% and 0.2% higher MAE, AEV and MAX errors for LPL values of 2 and 1 respectively. As the LPL increases above 2, the difference between BIC and precise model errors grows to about 5%.

Fig. 20 demonstrates the HBL simulation results when $\mathrm{WL} = 6, \mathrm{LPL} = 2$, and VBL is ineffective or zero. It shows that for $\mathrm{HBL} = 1$ and $\mathrm{HBL} = 2$, the mean and variance errors of the imprecise WPA are less than 0.8% and 1.2% higher than those of the precise WPA. Therefore, the $\mathrm{HBL} = 1$ is suitable to provide less than 1% difference.

With $\mathrm{WL} = 6, \mathrm{LPL} = 2$, and $\mathrm{HBL} = 1$, Fig. 21 shows the simulation results of the imprecise WPA with different VBLs. If we increase the VBL up to five, the mean and variance of the errors of the imprecise model increase by only 0.63% and 0.85% with respect to the precise WPA and the maximum error increases by 4%. As a conclusion, $\mathrm{WL} = 6, \mathrm{LPL} = 2, \mathrm{HBL} = 1$, and $\mathrm{VBL} = 5$ determine an optimum point of the BIC model in which, the error behavior of the BIC model tracks that of the precise model very closely (with less than 1% difference) while it is more efficient and has lower cost as shown next.

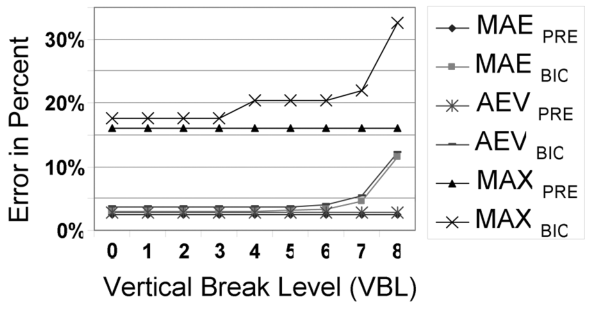


Fig. 21. MAE, AEV and MAX of the precise and BIC models for different VBLs while $\mathrm{WL} = 6, \mathrm{LPL} = 2$ and $\mathrm{HBL} = 1$.

TABLE II
THE PHYSICAL PROPERTIES AND ERROR FACTORS OF PRECISE AND BIC MODELS OF THE WPA DEFUZZIFICATION METHOD WITH DIFFERENT PARAMETER VALUES

	Parameter Values				**WPA Block Error Factors**			**WPA Block Synthesis Results**		
WPA Model Type	**WL**	**LPL**	**HBL**	**VBL**	**MAE**	**AEV**	**MAX**	**Area (# of gates)**	**Delay (ns)**	**Area × Delay**
P#1	6	-	-	-	2.43	2.80	15.9	1686	10.16	17129
P#2	7	-	-	-	2.45	3.04	14.2	2328	12.59	29309
B#1	6	2	1	5	3.05	3.64	20.3	994	7.48	7435
B#2	6	2	2	5	3.44	4.15	20.3	887	6.38	5659
B#3	6	2	1	6	3.29	3.91	20.3	811	6.59	5344
B#4	6	2	2	6	3.66	4.31	20.3	757	5.92	4481
B#5	7	2	1	6	2.54	3.01	15.1	1358	9.21	12507
B#6	7	2	2	6	2.53	2.84	15.2	1252	8.12	10166

After choosing some optimum points of the parameter space for the BIC model, these results are then used along with the synthesis results of the implied model to guide the designer through choosing the best precision-cost-performance trade-off as shown in the next sub-section.

C. Synthesis Results

Table II includes the synthesis results of the WPA precise and BIC models in some optimum points of the parameter space when synthesized with 0.13 μCMOS library cells using Leonardo Spectrum synthesis tool. The first column determines the type of the model ('P' for precise and 'B' for BIC models). The last three columns of the table show area, delay, and area $\times$ delay product of each model respectively. The error simulation results for each optimum point are also included in the table for clear comparison.

There are some considerations about the table values. According to Fig. 15, the defuzzification block critical path consists of a multiplier and an adder. Also its datapath consists of a multiplier; two adders and a divider. We set the divider aside because it works with much lower frequencies with respect to other components (the division frequency is related to the number of the output membership functions) and might be implemented in different styles with different area/delay trade-offs. To provide the numeric synthesis results, the precise model is built using an array multiplier with Ripple-Carry (RCA) merging adder, and two other RCA adders. In a similar way, the BIC model consists of a BAM with RCA merging adder and two LOAs (with RCA sub-adders).

The second and third rows of Table II include the experimental results of the precise model with $\mathrm{WL} = 6$ and 7 respectively. The next rows contain the results of the BIC model with different WL, LPL, HBL, and VBL values. The results show that all the imprecise models provide much better clock frequencies and implementation areas with respect to the precise model of the same WL while have slightly lower accuracy. As an example, the BIC#1 and BIC#3 instances provide less than 0.8% and 1.1% worse average and variance errors with respect to the precise model of the same WL (P#1) while have 35% and 54% higher clock frequencies. Also the hardware implementation of the precise model has an area $\times$ delay product of about 130% and 220% higher than BIC#1 and BIC#3 models respectively.

A more interesting observation is that the P#1 provides 37% and 68% higher area $\times$ delay product with respect to two BIC models with higher WL (B#5 and B#6) while their accuracy factors are quite similar.

VI. Conclusion

A new category of bio-inspired approximate hardware arithmetic blocks with enhanced implementation cost and performance is introduced. The BIC structures provide an estimation of the result instead of its precise value while are smaller, faster, and consume less power with respect to the conventional precise blocks. As case studies, two bio-inspired imprecise adder and multiplier structures are introduced. The error behavior and synthesis properties of these novel structures are discussed and presented. The applications of these BIC structures are then introduced in fuzzy and neural network fields. The simulation and synthesis results are provided that show these blocks can be exploited to efficiently implement the WPA defuzzification method as well as a three-layer computationally-intensive neural network for face recognition application at reduced cost.

References

[1] S. J. Ovaska, H. F. Van Landingham, and A. Kamiya, "Fusion of soft computing and hard computing in industrial applications: An overview," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 2, pp. 72–79, May 2002.

[2] L. A. Zadeh, "Soft computing and fuzzy logic," *IEEE Software*, vol. 11, no. 6, pp. 48–56, Nov. 1994.

[3] M. Hamzeh, H. R. Mahdiani, A. Saghafi, S. M. Fakhraie, and C. Lucas, "Computationally efficient active rule detection method: Algorithm and architecture," *Elsevier Fuzzy Sets Syst.*, vol. 160, no. 4, pp. 554–568, Feb. 2009.

[4] S. M. Fakhraie and K. C. Smith, *VLSI-Compatible Implementations for Artificial Neural Networks*. Norwell, MA: Kluwer, 1997.

[5] H. R. Mahdiani, A. Banaiyan, and S. M. Fakhraie, "Hardware implementation and comparison of new defuzzification techniques in fuzzy processors," in *Proc. IEEE ISCAS*, May 2006, pp. 4619–4622.

[6] B. Choi and K. Tipnis, "New components for building fuzzy logic circuits," in *Proc. Int. Conf. Fuzzy Syst. Knowledge Discovery*, Aug. 2007, vol. 2, pp. 586–590.

[7] Q. Cao, M. H. Lim, J. H. Li, Y. S. Ong, and W. L. Ng, "A context switchable fuzzy inference chip," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 4, pp. 552–567, Aug. 2006.

[8] I. del Campo, J. Echanobe, G. Bosque, and J. M. Tarela, "Efficient hardware/software implementation of an adaptive neuro-fuzzy system," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 3, pp. 761–778, Jun. 2008.

[9] S. G. Lee and J. D. Carpinelli, "VHDL implementation of very high-speed integer fuzzy controller," in *IEEE Int. Conf. Systems, Man Cybern.*, Oct. 2005, vol. 1, pp. 588–593.

[10] A. Kandel and G. Langholz, *Fuzzy Hardware: Architectures and Applications*. New York: Springer, 1998.

[11] G. Giustolisi, G. Palmisano, and G. Palumbo, "An efficient fuzzy controller architecture in SC technique," *IEEE Trans. Circuits Syst. II, Anal. Digit. Signal Process.*, vol. 49, no. 3, pp. 208–218, Mar. 2002.

[12] A. Rodriguez-Vazquez, R. Navas, M. Delgado-Restituto, and F. Vidal-Verdu, "A modular programmable CMOS analog fuzzy controller chip," *IEEE Trans. Circuits Syst. II, Anal. Digit. Signal Process.*, vol. 46, no. 3, pp. 251–265, Mar. 1999.

[13] K. Basterretxea, J. M. Tarela, I. del Campo, and G. Bosque, "An experimental study on nonlinear function computation for neural/fuzzy hardware design," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 266–283, Jan. 2007.

[14] Z. Xun, W. Peng, and J. Dongming, "VLSI architectures of domain adaptive fuzzy logic system," in *Proc. Int. Conf. ASIC*, Oct. 2005, vol. 1, pp. 257–260.

[15] P. Moerland and E. Fiesler, *Handbook of Neural Computation: Neural Network Adaptations to Hardware Implementations*. New York: Inst. of Physics/Oxford Univ. Press, 1997.

[16] T. J. Koickal, A. Hamilton, S. L. Tan, J. A. Covington, J. W. Gardner, and T. C. Pearce, "Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 60–73, Jan. 2007.

[17] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," in *Proc. IEE Circuits, Devices, Syst.*, Feb. 2004, vol. 151, no. 1, pp. 18–24.

[18] V. F. Koosh and R. M. Goodman, "Analog VLSI neural network with digital perturbative learning," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 5, pp. 359–368, May 2002.

[19] R. S. Schneider and H. C. Card, "Analog hardware implementation issues in deterministic Boltzmann machines," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 3, pp. 352–360, Mar. 1998.

[20] Y. L. Ju, I. M. Tsai, and S. Y. Kuo, "Quantum circuit design and analysis for database search applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2552–2563, Nov. 2007.

[21] H. Masahiko, T. Aoki, H. Morimitsu, and T. Higuchi, "Implementation of reaction-diffusion cellular automata," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 1, pp. 10–16, Jan. 2002.

[22] S. O'uchi, M. Fujishima, and K. Hoh, "An 8-qubit quantum-circuit processor," in *Proc. IEEE ISCAS*, May 2002, vol. 5, pp. 209–212.

[23] V. Beiu, S. Aunet, J. Nyathi, R. R. Rydberg, and W. Ibrahim, "Serial addition: Locally connected architectures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2564–2579, Nov. 2007.

[24] R. A. Blum, J. D. Ross, E. A. Brown, and S. P. DeWeerth, "An integrated system for simultaneous, multichannel neuronal stimulation and recording," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 12, pp. 2608–2618, Dec. 2007.

[25] E. W. Cheney and D. R. Kincaid, *Numerical Mathematics and Computing*, 6th ed. San Diego, CA: Brooks/Cole, 2008.

[26] H. R. Mahdiani, "Optimization and hardware modeling of the DMT engine of an ADSL modem for ASIC implementation," M.Sc. thesis, Elect. Comp. Eng. Dept., Univ. Tehran, Tehran, Iran, 2001.

[27] F. Frustaci, M. Lanuzza, P. Zicari, S. Perri, and P. Corsonello, "Designing high-speed adders in power-constrained environments," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 2, pp. 172–176, Feb. 2009.

[28] J. F. Lin, Y. T. Hwang, M. H. Sheu, and C. C. Ho, "A novel high-speed and energy efficient 10-transistor full adder design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1050–1059, May 2007.

[29] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1163–1170, Oct. 1993.

[30] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, 1997.

[31] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.

[32] K. H. Chen and Y. S. Chu, "A spurious-power suppression technique for multimedia/DSP applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 1, pp. 132–143, Jan. 2009.

[33] R. A. Patel, M. Benaissa, N. Powell, and S. Boussakta, "Novel power-delay-area-efficient approach to generic modular addition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1279–1292, Jun. 2007.

[34] J. M. Rabaey, A. Chandraksan, and B. Nikolic, *Digital Integrated Circuits, A Design Perspective*. Englewood Cliffs, NJ: Prentice Hall, 2003, pp. 591–594.

[35] N. H. E. Weste and D. Harris, *CMOS VLSI Design, A Circuits and Systems Perspective*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2004, pp. 693–696.

[36] H. Chang-Young, P. Hyoung-Joon, and K. Lee-Sup, "A low-power array multiplier using separated multiplication technique," *IEEE Trans. Circuits Syst. II, Analog. Digit. Signal Process.*, vol. 48, no. 9, pp. 866–871, Sep. 2001.

[37] H. J. Ko and W. S. Yu, "Finite wordlength effects analysis of a digital neural adaptive tracking controller for a class of nonlinear dynamical systems," in *Proc. Int. Joint Conf. Neural Networks*, May 2002, vol. 3, pp. 2466–2469.

[38] T. Mitchell, *Machine Learning*. New York: McGraw Hill, 1997 [Online]. Available: www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html, Ch. 4, Companion.

[39] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 10, pp. 3197–3208, Nov. 2008.

[40] S. Yoshizawa and Y. Miyanaga, "Use of a variable wordlength technique in an OFDM receiver to reduce energy dissipation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 9, pp. 2848–2859, Oct. 2008.

[41] A. Dastgheib and B. Murmann, "Calculation of total integrated noise in analog circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 10, pp. 2988–2993, Nov. 2008.

[42] R. O. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[43] S. Roychowdhury and W. Pedrycz, "A survey of defuzzification strategies," *Int. J. Intell. Syst.*, vol. 16, pp. 679–695, 2001.

[44] E. Van Broekhoven and B. De Baets, "A comparison of three methods for computing the center of gravity defuzzification," in *IEEE Int. Conf. Fuzzy Syst.*, Jul. 2004, vol. 3, no. 1, pp. 1537–1542.

[45] A. Patel and B. Mohan, "Some numerical aspects of center of area defuzzification method," *Fuzzy Sets and Syst.*, vol. 132, pp. 401–409, 2002.

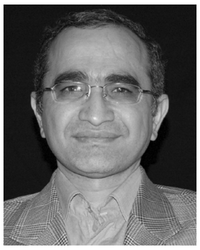
[46] A. Banaiyan, H. R. Mahdiani, and S. M. Fakhraie, "Cost-performance co-analysis in VLSI implementation of existing and new defuzzification methods," in *Proc. Int. Conf. Comput. Intell. Modeling, Contr. Automat.*, Nov. 2005, vol. 1, pp. 828–833.

Hamid Reza Mahdiani was born in Neyshabour, Iran, in 1976. He received the B.Sc. degree in computer engineering, and the M.Sc. degree in computer architecture from University of Tehran, Iran, in 1998 and 2001, respectively. Since 2004, he has been pursuing the Ph.D. degree in computer engineering with the School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran.

Since 2004, he has been a Research Associate and Lab Manager with the Silicon Intelligence and VLSI Signal Processing Laboratory, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran. From 2004, he has also worked as an instructor with Computer and Control Department, Sh. Abbaspour University of Technology, Tehran, Iran. His research interests include VLSI design of data communication and artificial intelligence systems, low-power and fault tolerant design in nano-level CMOS.

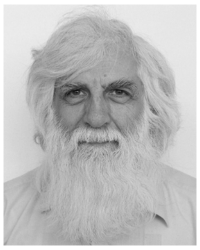
Ali Ahmadi received the B.Sc. degree in computer engineering from the University of Isfahan, Isfahan, Iran, in 2006, and the M.Sc. degree in computer architecture from the University of Tehran, Tehran, Iran, in 2009.

His research interests are fault-tolerant design of digital systems and Neural Networks, digital testing, and reconfigurable architectures.



Sied Mehdi Fakhraie was born in Dezfoul, Iran, in 1960. He received the M.Sc. degree in electronics from the University of Tehran, Tehran, Iran, in 1989, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1995.

Since 1995, he has been with the School of Electrical and Computer Engineering, University of Tehran, where he is now an Associate Professor. He is also the Director of Silicon Intelligence and the VLSI Signal Processing Laboratory. From September 2000 to April 2003, he was with Valence Semiconductor Inc. and has worked in Dubai, UAE, and Markham, Canada offices of Valence as Director of application-specific integrated circuit and system-on-chip (ASIC/SoC Design) and also technical lead of Integrated Broadband Gateway and Family Radio System baseband processors. During the summers of 1998, 1999, and 2000, he was a Visiting Professor at the University of Toronto, where he continued his work on efficient implementation of artificial neural networks. He is coauthor of the book *VLSI-Compatible Implementation of Artificial Neural Networks* (Kluwer, 1997). He has also published more than 120 reviewed conference and journal papers. He has worked on many industrial IC design projects including design of network processors and home gateway access devices, Digital Subscriber Line (DSL) modems, pagers, and one- and two-way wireless messaging systems, and digital signal processors for personal and mobile communication devices. His research interests include system design and ASIC implementation of integrated systems, novel techniques for high-speed digital circuit design, and system-integration and efficient VLSI implementation of intelligent systems.



Caro Lucas received the M.S. degree in electrical engineering from the University of Tehran, Tehran, Iran, in 1973 and the Ph.D. degree from the University of California, Berkeley, in 1976.

He is a Professor at the Center of Excellence for Control and Intelligent Processing, School of Electrical and Computer Engineering, University of Tehran, as well as a Researcher at the Intelligent Systems Research Faculty (ISRF), Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran. His research interests include biological computing, computational intelligence, uncertain systems, intelligent control, neural networks, multi-agent systems, data mining, business intelligence, financial modeling, and knowledge management.