A Real-Time Clustering Microchip Neural Engine

Teresa Serrano-Gotarredona and Bernabé Linares-Barranco

Centro Nacional de Microelectrónica (CNM), Dept. of Analog Design, Ed. CICA, Av. Reina Mercedes s/n, 41012 Sevilla, SPAIN, Phone: 34-5-4239923, Fax: 34-5-4624506,
E-mail: bernabe@cnm.us.es

Abstract

This paper presents an analog current-mode VLSI implementation of an unsupervised clustering algorithm. The clustering algorithm is based on the popular ART1 algorithm [1], but has been modified resulting in a more VLSI-friendly algorithm [2], [3] that allows a more efficient hardware implementation with simple circuit operators, little memory requirements, modular chip assembly capability, and higher speed figures. The chip described in this paper implements a network that can cluster 100 binary pixels input patterns into up to 18 different categories. Modular expansibility of the system is directly possible by assembling an $N \times M$ array of chips without any extra interfacing circuitry, so that the maximum number of clusters is $18 \times M$ and the maximum number of bits of the input pattern is $N \times 100$. Pattern classification and learning is performed in 1.8μs, which is an equivalent computing power of 4.4×10^9 connections per second plus connection-updates per second. The chip has been fabricated in a standard low cost 1.6μm double-metal single-poly CMOS process, has a die area of 1cm^2, and is mounted in a 120-pin PGA package. Although internally the chip is analog in nature, it interfaces to the outside world through digital signals, and thus has a true asynchronous digital behavior. Experimental chip test results are available, obtained through digital chip test equipment. Fault tolerance at the system level operation is demonstrated through the experimental testing of faulty chips.

I. Introduction

Two types of neural hardware engineers can be distinguished. The first designs "*general purpose*" hardware accelerators or systems that speed up neural algorithms running on conventional computers [4]-[12]. This kind of hardware allows considerable flexibility in the topology and operations of the neural systems. In this way algorithm researchers have a powerful tool to further develop neural algorithms and industry engineers have some attractive chips that significantly speed up their neural commercial products. The second type of hardware engineers are those who design a real-time system for a specific application. They must select the best-suited algorithm and map it into hardware. This achieves a close-to-optimum efficient hardware for a limited range of applications. The work described in this paper falls into this second category of hardware engineering. The specific application is real-time clustering of binary input patterns.

A clustering device is a device able to build categories from a collection of patterns. A **real-time** clustering device has to be able to do this at the speed of arrival of the patterns. There are some clustering algorithms [13]-[18] that need to be trained off-line to build the categories. For a real-time clustering device, however, it would be desirable to use an algorithm that can be trained on-line: if a new pattern arrives the algorithm updates it internal knowledge (instead of erasing all the accumulated knowledge and retrain with the old and new collection of patterns).

For the second type of neural hardware engineers, the issue of efficiently implementing in hardware a real size neural network is not a trivial task. Many neural network algorithms are available in the literature which have been developed, studied, and optimized for applications through computer and/or software based systems. Consequently, when designing a hardware realization, engineers face many problems like excessive

Submitted to *IEEE Trans. on VLSI Systems* on December 9, 1994. Accepted on November 9, 1995.

interconnectivity, high resolution of weights, high precision of operations, complicated operator requirements (e.g., integrals and derivatives), high number of neurons required for a real-world application, etc. Many times some of these requirements can be relaxed, the topology modified, or the operations simplified, with no significant deterioration of global operation of the neural system but with a considerable boost in the hardware performance. Modifying neural algorithms to make them more VLSI-friendly and produce more efficient hardware should be a common practice among neural hardware engineers of the second type [19]-[22]. After selecting an appropriate neural algorithm the next step consists of studying how far the algorithm can be simplified without performance degradation. The simplifications have to be hardware-oriented, so that the final combination of "*theoretical algorithm*" + "*hardware circuit technique*" results in a high performance real time system. The success of the hardware system depends on the selection of the algorithm, the selection of a powerful circuit design technique, and how the algorithm is modified to efficiently "*marry*" the circuit technique resulting in an optimum performance final system.

In the case of our application, real-time binary patterns clustering, we chose the ART1 algorithm mainly due to the attractive hardware-oriented properties, as well as the theoretical computational properties that will be highlighted below. We also chose to slightly modify the mathematical ART1 algorithm to obtain more efficient hardware. This modification (described in the next Section) allows the use of simpler operations while preserving all the computational properties of the original ART1 architecture [2], [3]. As an extra bonus, the hardware circuit introduces a significant speed improvement as it automatically parallels the sequential ART search process [1] inherent in the mathematical neural algorithm.

The advantageous features of the ART1 algorithm are described next, as well as different possible mathematical levels of description:

A. Computational Properties of the ART1 Algorithm:

From a purely algorithmic point of view, the ART1 architecture is capable of learning, in an unsupervised way, recognition codes in response to arbitrary orderings of arbitrarily many and complex binary input patterns. This architecture has a collection of interesting computational properties [1]:

- *Self-Scaling:* The self-scaling property discovers critical features in a context-sensitive way.
- *Vigilance or Variable Coarseness:* There is a vigilance parameter ($0 < \rho \leq 1$) that allows tuning the coarseness of the categories to learn.
- *Subset and Superset Direct Access:* The system is able to classify a new input pattern as belonging to either a subset or a superset category, depending on global similarity criteria. No restrictions on input orthogonality or linear predictability are needed.
- *Stable Category Learning:* In response to an arbitrary list (finite or infinite) of binary input patterns, learning is assured to self-stabilize within a finite number of learning trials.
- *Biasing the Network to form New Categories:* There is a parameter that can bias the tendency of the system to code unfamiliar patterns into new categories, independent of the vigilance parameter.
- *On-Line Learning:* The ART1 algorithm learns as it performs, as opposed to other algorithms, where first the algorithm must be trained and second, it can be used in an application. The ART1 algorithm can incorporate new knowledge as it is being used. This property makes ART1 an excellent candidate for real-time clustering.

- *Capturing Rare Events:* ART1 is able to identify and build clusters of events that appear with a very low frequency. Even if an event corresponding to a clearly distinct cluster appears only once, ART1 is able to detect it while building and preserving the corresponding cluster or category.

B. Hardware-Oriented Attractive Properties of the ART1 Algorithm

In performance comparison of hardware implementations, a common figure of merit is the number of interconnections per second. More refined figures have recently been proposed that include resolution and precision [23]. However, these figures would be reasonably fair criteria for the first type of hardware engineering mentioned above, the *general-purpose* one. In order to compare hardware systems of the second type, the *specific-application* neural hardware, some global figure must be used that evaluates the overall system performance. Usually this figure will be application dependent. In our case, since we are concerned with a real-time clustering application of binary input patterns, an appropriate figure of merit might be

$$ppc/s = \frac{\text{number of } \mathbf{p}\text{atterns processed}}{\mathbf{s}\text{econds}} \times \mathbf{p}\text{ixels} \times \mathbf{c}\text{ategories} \tag{1}$$

where,

- **number of patterns processed/second** is the speed at which patterns are classified and learned (including the number of learning trials required). This speed generally depends on the patterns themselves, and on the knowledge already stored in the system. Therefore, this speed can be given as an average or as the slowest case measured.
- **pixels** is the maximum number of pixels of the input patterns.
- **categories** is the maximum number of categories the system is able to form.

As we will see later in the Section on experimental results, the chip described in this paper is able to cluster up to 18 different categories of binary patterns with 100 pixels, while classifying and learning each pattern in less than 1.8µs. Since ART1 learns on-line, 1 iteration of input patterns presentations provides the system with sufficient knowledge to perform properly[1]. This results in a *ppc/s* of

$$ppc/s = \frac{n \text{ patterns}}{1 \text{ iteration} \times n \text{ patterns} \times 1.8\mu s} \times 100 \text{ pixels} \times 18 \text{ categories} = 1.0 \times 10^{9} ppc/s \tag{2}$$

If we would like to obtain the same performance using *Backpropagation* based hardware, and assuming the network would learn with 10,000 iterations of patterns presentations, this means that a speed of 180*ps* would be needed for each pattern classification and corresponding weights update. Assuming this task could be performed with a Backpropagation network with 100 input neurons, 5 hidden-layer neurons, and 5 output neurons[2] (which means a total of $100 \times 5 + 5 \times 5 = 525$ interconnections), and that the speed of feedforward classification is the same as for feedback learning, hardware able to perform

$$\frac{2 \times 525 \text{ connections}}{180ps} = 5.83 \times 10^{12} \text{connections/s plus connection-updates/s} \tag{3}$$

1. The input patterns set can be iterated several times to stabilize the internal weights, but this is not necessary for the system to start working.

2. Optimistically, a backpropagation net with 5 output nodes might be able to code up to 2^5 categories.

would be needed. For the chip described in this paper, since it is based on the powerful ART1 algorithm, the above performance can be achieved with a hardware of only 4.4×10^9 *connections/s* plus *connections-updates/s*, as discussed in Section IV.B.3.

Note that the Backpropagation algorithm is not appropriate for clustering applications, and comparing it against ART1 is slightly unfair. There are other algorithms available in the literature that have been developed specially for clustering applications [13]-[18]. However, they usually do not provide all the computational properties mentioned in subsection *A* previously, specially the "*On-Line Learning*" property which is crucial for real-time clustering, or they present serious difficulties when mapped into hardware.

Another hardware attractive feature that an ART1 based implementation offers with respect to others, is that the interconnection weights do not have to be analog, as shown in the next Section. Most of the neural algorithms reported in the literature require a real-valued set of weights defined within a certain interval. These weights can be discretized in a number of digital steps, but the granularity required for proper operation of the system is usually very fine (around 16-bits for the Back-Propagation algorithm [24]). Even worse, in some cases the granularity requirements become more severe as the size of the system increases. For example, in a BAM system [25] of $N \times M$ neurons, storage capacity has been heuristically estimated to be around $n_p = (N \times M)^{1/4}$ [26], where n_p is the average maximum number of patterns that can be stored. The resolution required by the interconnection weights in this case is at least $n_p + 1$. In the chip described in this paper, since it is based on the ART1 algorithm and requires only binary-valued weights, the resolution of the weights is not affected by the size nor the storage capacity of the system. This, and the non necessity of analog weights is one of the most hardware attractive features of the ART1 algorithm.

Another consideration to take into account during the design of a hardware system is how it scales up with size and performance. We have already mentioned that some neural systems need to increase their weight resolution as they scale up. Another feature is how their size and interconnectivity scale up with pattern size or storage capacity. For an ART1 based system, the number of neurons *N* in the bottom layer is the number of pixels of the patterns, the number of neurons *M* in the top layer is the maximum number of categories, and $N \times M$ is the number of synapses. This system scales up linearly with storage capacity (*M*) and input pixels (*N*). For a BAM system, for example, the size scales quadratically with the storage capacity and the number of pixels.

Section V will present other scaling considerations, more directly related to the hardware technique selected. In the case of an analog hardware, random and systematic errors due to fabrication process variations will appear. A neural network can usually cope very well with random errors, even if the size of the system increases. However, systematic errors may accumulate as the system increases and may render the complete network useless as it scales up. The chosen circuit technique must be either insensitive to the accumulation of systematic errors, or allow for some kind of calibration technique to overcome them.

C. Description Levels of the ART1 Algorithm:

In the original ART1 paper [1] the architecture is mathematically described by sets of Short Term Memory (STM) and of Long Term Memory (LTM) time domain nonlinear differential equations. A valid assumption also done by Carpenter and Grossberg, is to make the STM differential equations settle instantaneously to their corresponding steady state, and consider only the dynamics of the LTM differential equations. In this case, the STM differential equations must be substituted by nonlinear algebraic equations that describe the corresponding steady state of the system. Furthermore, Carpenter and Grossberg also introduced the fast learning mode of the ART1 architecture, in which the LTM differential equations are also substituted by their corresponding steady-state nonlinear algebraic equations. Thus the ART1 architecture originally modelled as a dynamically evolving collection of neurons and synapses governed by time-domain nonlinear differential equations, can be behaviorally modelled as the sequential application of nonlinear algebraic equations: an input pattern is given, the corresponding STM steady state is computed through the STM algebraic equations, and the system weights are updated using the corresponding LTM algebraic equations.

At this point three different levels of ART1 implementations (both in software or in hardware) can be distinguished:

Type-1: *Full Model Implementation:* Both STM and LTM time-domain differential equations are realized. This implementation is the most expensive (both in software and in hardware), and requires a large amount of computational power.

Type-2: *STM Steady-State Implementation:* Only the LTM time-domain differential equations are implemented. The STM behavior is governed by nonlinear algebraic equations. This implementation requires less resources than the previous one. However, proper sequencing of STM events must be introduced artificially, which is architecturally implicit in the *Type-1* implementation.

Type-3: *Fast Learning Implementation:* This implementation is computationally the least expensive. In this case, STM and LTM events must be artificially sequenced.

Regarding hardware implementations of the ART1 architecture, several attempts have been reported in the literature. Ho et al. suggested a *Type-1* implementation [27]. Tsay and Newcomb proposed a CMOS circuit technique that would realize a partial *Type-2* implementation [28]; Wunsch et al. [29] have built optical-based *Type-3* implementations; this paper presents a CMOS VLSI *Type-3* circuit.

The next Section explains how we slightly modified the ART1 algorithm to make it more VLSI-friendly. Section III describes the circuit implementation of the so-modified ART1 algorithm using analog current-mode circuit design techniques. Experimental results of an actual prototype chip are given in Section IV. Section V highlights some potential improvements that would help to make this chip an industry-ready commercial chip, and finally, some conclusions are made in Section VI.

II. A VLSI-friendly ART1 Algorithm

Let us start describing the *Type-3* model of the original ART1 architecture. The ART1 topology is shown in Fig. 1, and consists of two layers: layer *F1* is the input layer and has N nodes (one for each binary "pixel" of the input pattern), and layer *F2* is the category layer. Each node in the *F2* layer represents a "cluster" or "category". In this layer only one node will become active after presentation of an input pattern $\mathbf{I} \equiv (I_1, I_2, \ldots, I_N)$. The *F2* layer category that will become active is that which most closely represents the input pattern **I**. If no preexisting category is satisfactory for a given input pattern, a new category will be formed. Each *F1* node x_i is connected to all *F2* nodes y_j through bottom-up connections of weights[3] z_{ij}^{bu}, so that the input received by each *F2* node y_j is given by

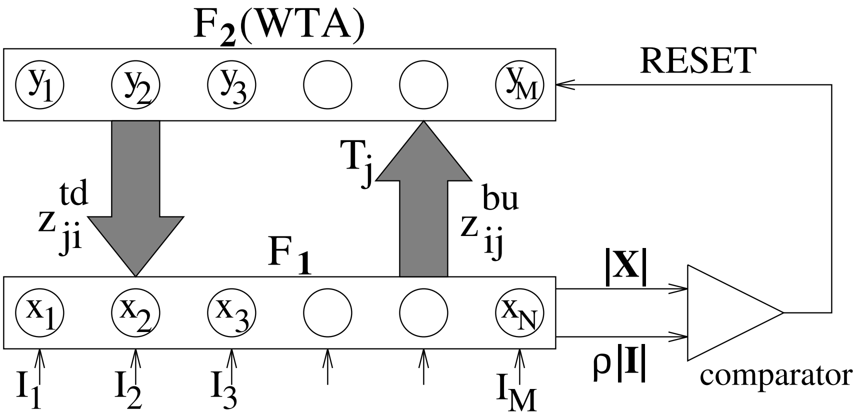


Fig. 1: Simplified block diagram of the architecture of a *Type-3* ART1 system

$$T_j = \sum_{i=1}^{N} z_{ij}^{bu} I_i \tag{4}$$

Layer *F2* acts as a Winner-Take-All network, so that all nodes y_j remain inactive, except that which receives the largest bottom-up input T_j,

$$y_j = \begin{cases} 1 & \text{if } T_j = max_k \{T_k\} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Once an *F2* winning node arises, a top-down pattern is activated through the top-down weights[4] z_{ji}^{td}. Let us call this top-down pattern $\mathbf{X} \equiv (X_1, X_2, \ldots, X_N)$. The resulting vector **X** is given by the equation,

$$X_i = I_i \sum_j z_{ji}^{td} y_j \tag{6}$$

Since only one y_j is active, let us call this winning *F2* node y_J, so that y_j=0 if $j \neq J$ and y_J=1. In this case we can state

$$X_i = I_i z_{Ji}^{td} \qquad or \qquad \mathbf{X} = \mathbf{I} \cap \mathbf{z}_J^{td} \tag{7}$$

where $\mathbf{z}_J^{td} \equiv \left(z_{1J}^{td}, z_{2J}^{td}, \ldots z_{NJ}^{td} \right)$. This top-down template will be compared with the original input pattern **I** according to a predetermined *vigilance* criterion, tuned by the *vigilance parameter* $0 < \rho \leq 1$, so that two alternatives may occur:

a) If[5] $\rho|\mathbf{I}| \leq \left|\mathbf{I} \cap \mathbf{z}_J^{td}\right|$ the active category J is accepted and the system weights will be updated to incorporate this new knowledge.

---

3. Bottom-up weights z_{ij}^{bu} may take any real value in the interval [0,*K*], where $K = \frac{L}{L-1+N}$, and $L > 1$ [1].

4. In the *Fast Learning (Type-3)* model top-down weights z_{ji}^{td} may take only the values ‘0’ or ‘1’.

5. The notation |**a**| represents the cardinality of vector **a**, i.e., $|\mathbf{a}| = \sum_{i=1}^{N} |a_i|$.

b) If $\rho|\mathbf{I}| > \left|\mathbf{I} \cap \mathbf{z}_J^{td}\right|$ the active category J is not valid for the actual value of the *vigilance parameter* ρ. In this case y_J will be deactivated (reset) making $T_J = 0$, so that another y_j node will become active through the Winner-Take-All action of the *F2* layer.

Learning takes place when an active *F2* node is accepted by the vigilance criterion. The weights will be updated according to the following algebraic equations,

$$\left.z_{iJ}^{bu}\right|_{new} = \frac{L}{L-1+\left|\left.\mathbf{z}_J^{td}\right|_{old} \cap \mathbf{I}\right|} X_i = \frac{L}{L-1+\left|\left.\mathbf{z}_J^{td}\right|_{old} \cap \mathbf{I}\right|} I_i \left.z_{Ji}^{td}\right|_{old}$$
$$\left.z_{Ji}^{td}\right|_{new} = X_i = I_i \left.z_{Ji}^{td}\right|_{old} \tag{8}$$

or using vector notation

$$\left.\mathbf{z}_J^{bu}\right|_{new} = \frac{L\mathbf{I} \cap \left.\mathbf{z}_J^{td}\right|_{old}}{L-1+\left|\mathbf{I} \cap \left.\mathbf{z}_J^{td}\right|_{old}\right|}$$
$$\left.\mathbf{z}_J^{td}\right|_{new} = \mathbf{I} \cap \left.\mathbf{z}_J^{td}\right|_{old} \tag{9}$$

where parameter L has to be larger than '1' [1]. Note that only the weights of the connections touching the *F2* winning node y_J are updated. Therefore, operation of the *Type-3* (or *Fast Learning*) implementation of the ART1 architecture is described by the algorithm depicted in Fig. 2(a).

From a hardware implementation point of view, one of the first issues that comes into consideration is that there are two templates of weights to be built. The set of bottom-up weights z_{ij}^{bu}, each of which must store a real value belonging to the interval [0,K], and the set of top-down weights z_{ji}^{td}, each of which stores either the value '0' or '1'. The physical implementation of the bottom-up template memory presents the first hardware difficulty, because their weights need either an analog or a digital memory with sufficient bits per weight so that the digital discretization does not affect the system performance. However, looking at eqs. (8) it can be seen that the bottom-up set $\{z_{ij}^{bu}\}$ and the top-down set $\{z_{ji}^{td}\}$ contain the same information: each of these sets can be fully computed by knowing the other set. It can be seen that the bottom-up set z_{ij}^{bu} is a normalized version of the top-down set z_{ji}^{td}. Therefore, from a hardware implementation point of view it would be desirable to physically implement only a binary valued set (one bit per weight) and let the hardware do the normalization of the bottom-up weights during the computation of $\{T_j\}$. This way, the two sets $\{z_{ij}^{bu}\}$ and $\{z_{ji}^{td}\}$ can be substituted by a single binary valued set $\{z_{ij}\}$, and eq. (4) modified to take into account the normalization effect of the original bottom-up weights[6],

$$T_j = \frac{LT_{Aj}}{L-1+T_{Bj}} = \frac{L\sum_{i=1}^{N} z_{ij}I_i}{L-1+\sum_{i=1}^{N} z_{ij}} = \frac{L|\mathbf{z}_j \cap \mathbf{I}|}{L-1+|\mathbf{z}_j|} \tag{10}$$

---

6. Note that we are using the notation $\mathbf{z}_j$ to represent the vector $(z_{1j}, z_{2j}, \ldots z_{Nj})$.

Considering this minor “implementation” modification the algorithm of Fig. 2(a) would be transformed into that depicted in Fig. 2(b). The system level performance of the algorithms described by Fig. 2(a) and Fig. 2(b) are identical. There is no difference in the behavior between the two diagrams, and the one in Fig. 2(b) offers more attractive features from a hardware (as well as software) implementation point of view.

However, in Fig. 2(b) an extra division operation, $T_j = (LT_{Aj})/(L-1+T_{Bj})$, need be performed for each node in the *F2* layer. This is an expensive hardware operation and would probably constitute a performance bottleneck in the overall system for both analog and digital circuit implementations. If possible, it would be very desirable to avoid this division operation. In [2] and [3] we show that this division operation can be substituted by a substraction operation, while preserving all the computational properties of the original ART1 algorithm. For some sequence of patterns a different behavior can be observed with respect to the original ART1, but the overall clustering behavior is still equivalent. Mathematically, the input to the *F2* layer is now,

$$T_j = L_A T_{Aj} - L_B T_{Bj} + L_M \tag{11}$$

where L_A and L_B are positive parameters that play the role of the original *L* (and *L*–1) parameter. The condition $L_A > L_B$ must be imposed for proper system operation [2], [3]. $L_M > 0$ is a constant parameter needed to assure that $T_j \geq 0$, for all possible values of T_{Aj} and T_{Bj}.

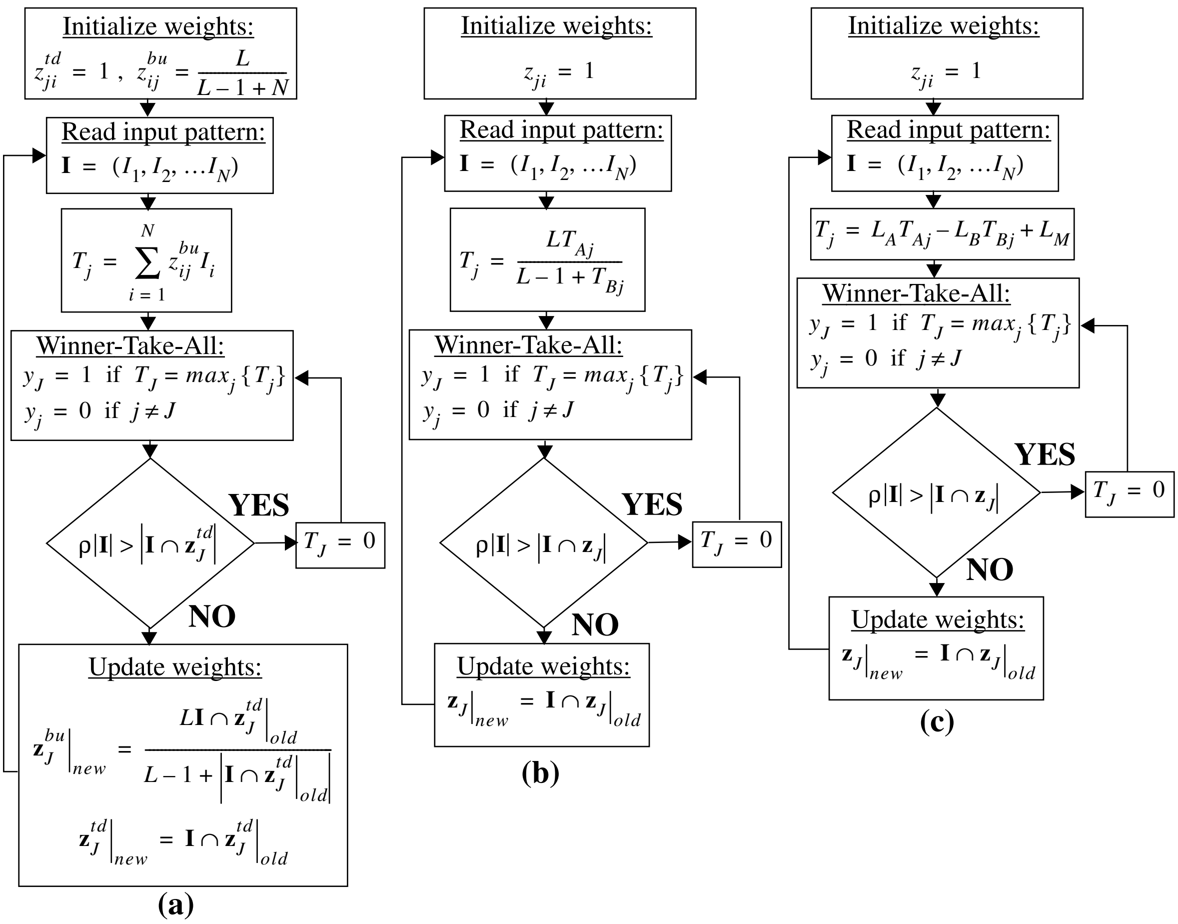


Fig. 2: *Type-3* implementation algorithms of the ART1 architecture: (a) original ART1, (b) ART1 with a single binary valued weights template, (c) modified VLSI-friendly ART1

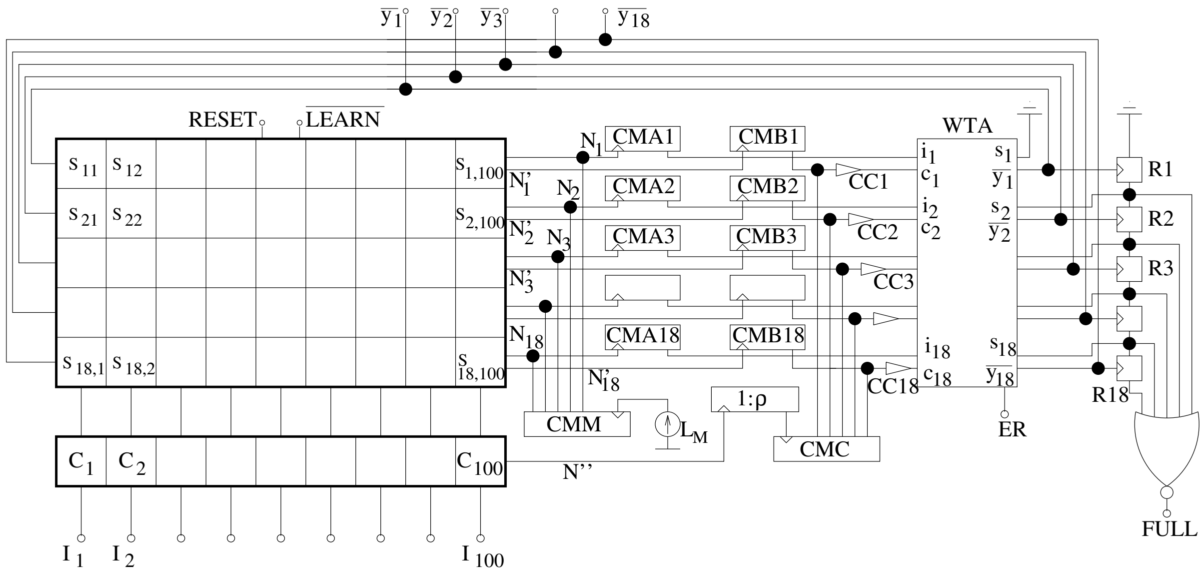


Fig. 3: Hardware Block Diagram for the modified VLSI-friendly ART1 algorithm

Replacing a division operation with a substraction one is a very important hardware simplification with significant performance improvement potential. Fig. 2(c) shows the final VLSI-friendly *Type-3* ART1 algorithm, which has been mapped into hardware, as described in the next Section.

III. Circuit Description

The operations in Fig. 2(c) that need to be implemented are the following:

- Generation of the terms T_j. Since z_{ij} and I_i are binary valued (0 or 1), "binary multiplication" and addition/substraction operations are required.
- Winner-Take-All (WTA) operation to select the maximum T_j term.
- Comparison of the term $\rho|\mathbf{I}|$ with $|\mathbf{I} \cap \mathbf{z}_J|$.
- Deselection of the term T_J if $\rho|\mathbf{I}| > |\mathbf{I} \cap \mathbf{z}_J|$.
- Update of weights.

The first three operations require a certain amount of precision, while the last two operations are not precise. We intended to obtain a precision between 1 and 2% (equivalent to 6-bits) for our circuit, while handling input patterns of up to 100 binary pixels. Fig. 3 shows a possible hardware block diagram that would physically implement the algorithm of Fig. 2(c). The circuit consists of an 18×100 array of *synapses* $S_{11}, S_{12}, \ldots, S_{18,\,100}$, a 1×100 array of *controlled current sources* $C_1, C_2, \ldots, C_{100}$, two 1×18 arrays of unity-gain current mirrors $CMA1, \ldots, CMA18$, $CMB1, \ldots CMB18$, a 1×18 array of current comparators $CC1, \ldots, CC18$, an 18-input WTA circuit, two 18-output unity-gain current mirrors *CMM* and *CMC*, and an adjustable-gain ($0 < \rho \leq 1$) current mirror. Registers $R1, \ldots, R18$ and the NOR gate are optional, and their function is explained later.

Each synapse receives two input signals $\bar{y}_j$ and I_i, has two global control signals RESET and $\overline{\text{LEARN}}$, stores the value of z_{ij}, and generates two output currents:

- the first goes to the input of current mirror *CMAj* and is $L_A z_{ij} I_i - L_B z_{ij}$.

- the second goes to the input of current mirror *CMBj* and is $L_A z_{ij} I_i$.

All synapses in the same row j ($S_{j1}, S_{j2}, \ldots, S_{j,100}$) share the two nodes ($N_j$ and N'_j) into which the currents they generate are injected. Therefore, the input of current mirror *CMAj* receives the current

$$T_j = L_A \sum_{i=1}^{100} z_{ij} I_i - L_B \sum_{i=1}^{100} z_{ij} + L_M = L_A \left|\mathbf{I} \cap \mathbf{z}_j\right| - L_B \left|\mathbf{z}_j\right| + L_M \qquad (12)$$

while the input of current mirror *CMBj* receives the current

$$L_A \sum_{i=1}^{100} z_{ij} I_i = L_A \left|\mathbf{I} \cap \mathbf{z}_j\right| \qquad (13)$$

Current L_M, which is replicated 18 times by current mirror *CMM* has an arbitrary value as long as it assures that the terms T_j are positive.

Each element of the array of *controlled current sources* C_i has one input signal I_i and generates the current $L_A I_i$. All elements C_i share their output node, so that the total current they generate is $L_A|\mathbf{I}|$. This current reaches the input of the adjustable gain ρ current mirror, and is later replicated 18 times by current mirror *CMC*.

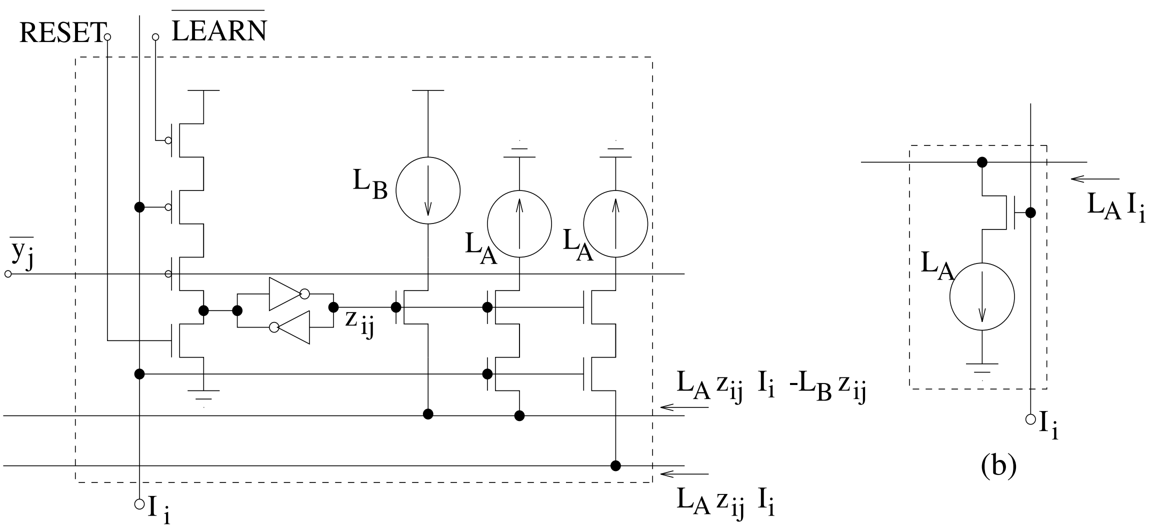
Each of the 18 current comparators *CCj* receives the current $L_A\left|\mathbf{I} \cap \mathbf{z}_j\right| - L_A \rho |\mathbf{I}|$ and compares it against zero. If this current is positive, the output of the current comparator falls, but if the current is negative the output rises. Each current comparator *CCj* output controls input c_j of the WTA. If c_j is high the current sunk by the WTA input i_j (which is T_j) will not compete for the winning node. On the contrary, if c_j is low, input current T_j will enter the WTA competition. The outputs of the WTA $\bar{y}_j$ are all high, except for that which receives the largest $\bar{c}_j T_j$: such output, denominated $\bar{y}_J$, will fall.

Now we can describe the operation of the circuit in Fig. 3. All synaptic memory values z_{ij} are initially set to ‘1’ by the RESET signal. Once the input vector $\mathbf{I}$ is activated, the 18 rows of synapses generate the currents $L_A\left|\mathbf{I} \cap \mathbf{z}_j\right| - L_B\left|\mathbf{z}_j\right|$ and $L_A\left|\mathbf{I} \cap \mathbf{z}_j\right|$, and the row of controlled current sources $C_1, \ldots, C_{100}$ generates the current $L_A|\mathbf{I}|$. Each current comparator *CCj* will prevent current $T_j = L_A\left|\mathbf{I} \cap \mathbf{z}_j\right| - L_B\left|\mathbf{z}_j\right| + L_M$ from competing in the WTA if $\rho|\mathbf{I}| > \left|\mathbf{I} \cap \mathbf{z}_j\right|$. Therefore, the effective WTA inputs are $\{\bar{c}_j T_j\}$, from which the WTA chooses the maximum, making the corresponding output $\bar{y}_J$ fall. Once $\bar{y}_J$ falls, and assuming the synaptic control signal $\overline{\text{LEARN}}$ is low, all z_{iJ} values will change from ‘1’ to ‘I_i’.

Note that initially (when all $z_{ij} = 1$),

$$\bar{c}_j T_j = L_A|\mathbf{I}| - L_B N + L_M \qquad (N{=}100) \qquad \forall j \qquad (14)$$

This means that the winner will be chosen among 18 equal competing inputs, basing the election on mismatches due to random process parameter variations of the transistors. Even after some categories are learned, there will be a number of uncommitted rows ($z_{1j} = \ldots = z_{100,j} = 1$) that generate the same competing current of eq. (14). The operation of a WTA circuit in which there are more than 1 equal and winning inputs becomes more difficult and in the best case, renders slower operation. To avoid these problems 18 D-registers, $R1, \ldots, R18$, might be added. Initially these registers are set to ‘1’ so that the WTA inputs



(a)

Fig. 4: (a) Details of Synapse Circuit S_{ij}, (b) Details of Controlled Current Source Circuit C_i

$s_2, \ldots, s_{18}$ are high. Inputs $s_1, \ldots, s_{18}$ have the same effect as inputs $c_1, \ldots, c_{18}$: if s_j is high T_j does not compete for the winner, but if s_j is low T_j enters the WTA competition. Therefore, initially only $\bar{c}_1 T_1$ competes for the winner. As soon as $\bar{y}_1$ rises once, the input of register *R*1 (which is ‘0’) is transmitted to its output making $s_2 = 0$. Now both $\bar{c}_1 T_1$ and $\bar{c}_2 T_2$ will compete for the winner. As soon as $\bar{c}_2 T_2$ wins once, the input of register *R*2 is transmitted to its output making $s_3 = 0$. Now $\bar{c}_1 T_1$, $\bar{c}_2 T_2$, and $\bar{c}_3 T_3$ will compete, and so on. If all available *F2* nodes ($y_1, \ldots, y_{18}$) have won once, the “FULL” signal rises, advising that all *F2* nodes are storing a category. The WTA control signal “ER” enables operation of the registers.

A. Synaptic Circuit and Controlled Current Sources:

The details of a synapse S_{ij} are shown in Fig. 4(a). It consists of three current sources (two of value L_A and one of value L_B), a two-inverter loop (acting as a Flip-Flop), and nine MOS transistors working as switches. As can be seen in Fig. 4(a) each synapse generates the currents $L_A z_{ij} I_i - L_B z_{ij}$ and $L_A z_{ij} I_i$. The RESET control signal sets z_{ij} to ‘1’. Learning is performed by making z_{ij} change from ‘1’ to ‘0’ whenever $\overline{\mathrm{LEARN}} = 0$, $\bar{y}_j = 0$, and $I_i = 0$.

Fig. 4(b) shows the details of each controlled current switch C_i. If $I_i = 0$ no current is generated, while if $I_i = 1$, the current L_A is provided.

B. Winner-Take-All (WTA) Circuit:

Fig. 5 shows the details of the WTA circuit. It is based on Lazzaro’s WTA [30], which consists of the array of transistors *MA* and *MB*, and the current source I_{BIAS}. Transistor *MC* has been added to introduce a cascode effect and increase the gain of each cell. Transistors *MX*, *MY*, and *MZ* transform the output current into a voltage, which is then inverted to generate $\bar{y}_j$. Transistor *MT* disables the cell if c_j is high, so that the input current T_j will not compete for the winner. Transistors *MS* and *ME* have the same effect as transistor *MT*: if signals ER and s_j are high, T_j will not compete.

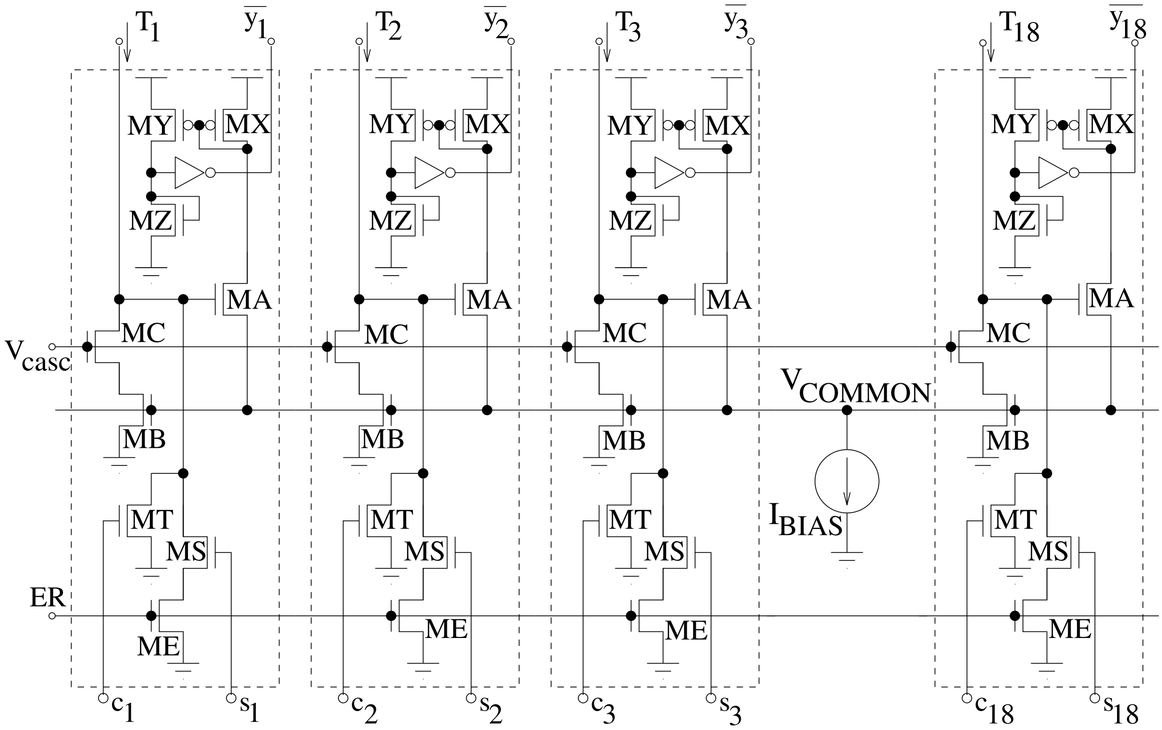


Fig. 5: Circuit Schematic of Winner-Take-All (WTA) Circuit

C. Current Comparators:

The circuit used for the current comparators is shown in Fig. 6(a). Such a comparator forces an input voltage approximately equal to the inverters trip voltage, has extremely high resolution (less than 1pA), and can be extremely fast (in the order of 10-20ns for input around 10μA) [31].

D. Current Mirrors:

Current Mirrors $CMA1, \ldots, CMA18, CMB1, \ldots, CMB18, CMM, CMC$, and the ρ-gain mirror have been laid out using common centroid layout techniques to minimize matching errors and keep the 6-bit precision of the overall system. For current mirrors $CMA1, \ldots, CMA18$ and $CMB1, \ldots, CMB18$ a special topology has been used, shown in Fig. 6(b) [32]. This topology forces a constant voltage V_D at its input node, thus producing a virtual ground in the output nodes of all synapses, which reduces channel length modulation distortion improving matching between the currents generated by all synapses. In addition, the topology of Fig. 6(b) presents a very wide current range with small matching errors [32].

The adjustable gain ρ current mirror also uses this topology, as shown in Fig. 6(c). Transistor $M0$ has a geometry factor (W/L) 10 times larger than transistors $M1, \ldots, M10$. Transistors $MR1, \ldots, MR10$ act as switches (controlled by signals $r_1, \ldots, r_{10}$), so that the gain of the current mirror can be adjusted between

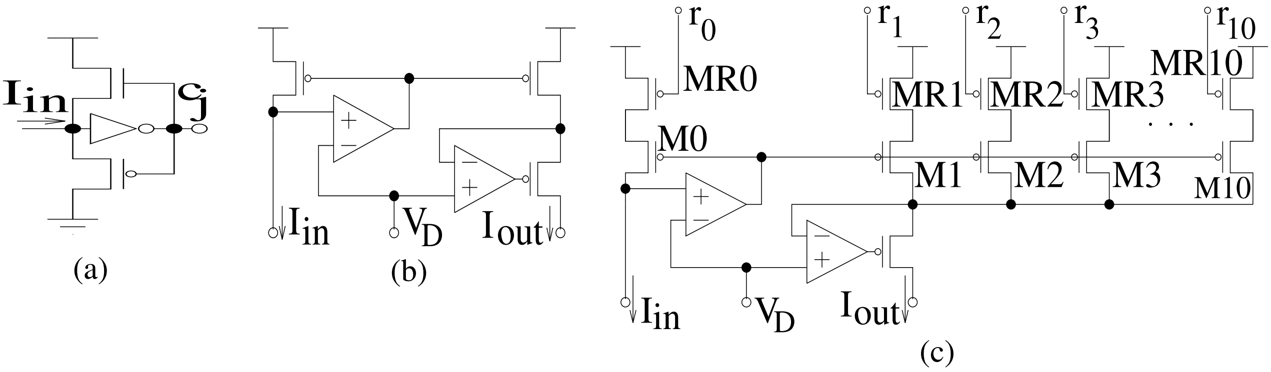


Fig. 6: (a) Circuit Schematic of Current Comparator. (b) Circuit Schematic of Active-Input Regulated-Cascode Current Mirror. (c) Circuit Schematic for Adjustable Gain ρ Current Mirror

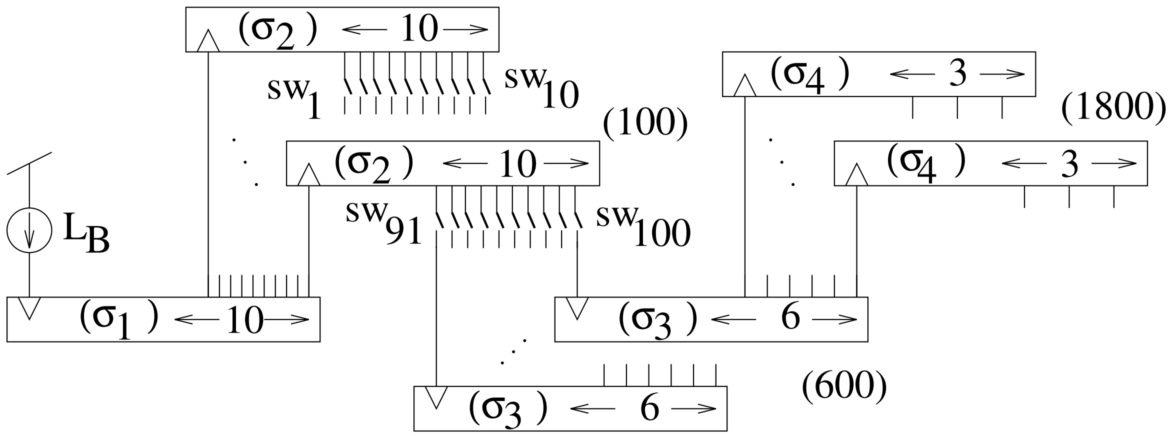


Fig. 7: Cascade of Current Mirrors for low Mismatching

$\rho = 0.0$ to $\rho = 1.0$ in steps of 0.1, while maintaining $r_0 = 0$. By making r_0 higher than 0 Volts, ρ can be fine tuned.

E. Synaptic Current Sources:

The current sources L_A and L_B inside each synapse S_{ij} and controlled current sources C_i have to match within approximately 1% to keep the system 6-bit precision. There is a total of $100 \times 18 \times 2 + 100 = 3700$ L_A current sources and $100 \times 18 = 1800$ L_B current sources spread over a die area of $1 cm^2$ which have to match within 1%. For such distances, number of current sources, and reasonable current values, a spread of 10% in the currents would be an optimistic estimate. However, a single current mirror, with a reduced number of outputs (like 10), a reasonable transistor size (like $40\mu m \times 40\mu m$), a moderate current (around $10\mu A$), and using common centroid layout techniques can be expected to have a mismatch error standard deviation σ_q of less than 1% [33]. By cascading several of these current mirrors in a tree-like fashion as is shown in Fig. 7 (for current sources L_B), a high number of current sources (copied from a single common reference) can be generated with a mismatch equal to

$$\sigma_{Total} = \sigma_1 + \sigma_2 + \ldots + \sigma_q \tag{15}$$

Each current mirror stage introduces an error σ_k. This error can be reduced by increasing the transistor areas of the current mirrors. Since the last stage q has a higher number of current mirrors, it is important to keep

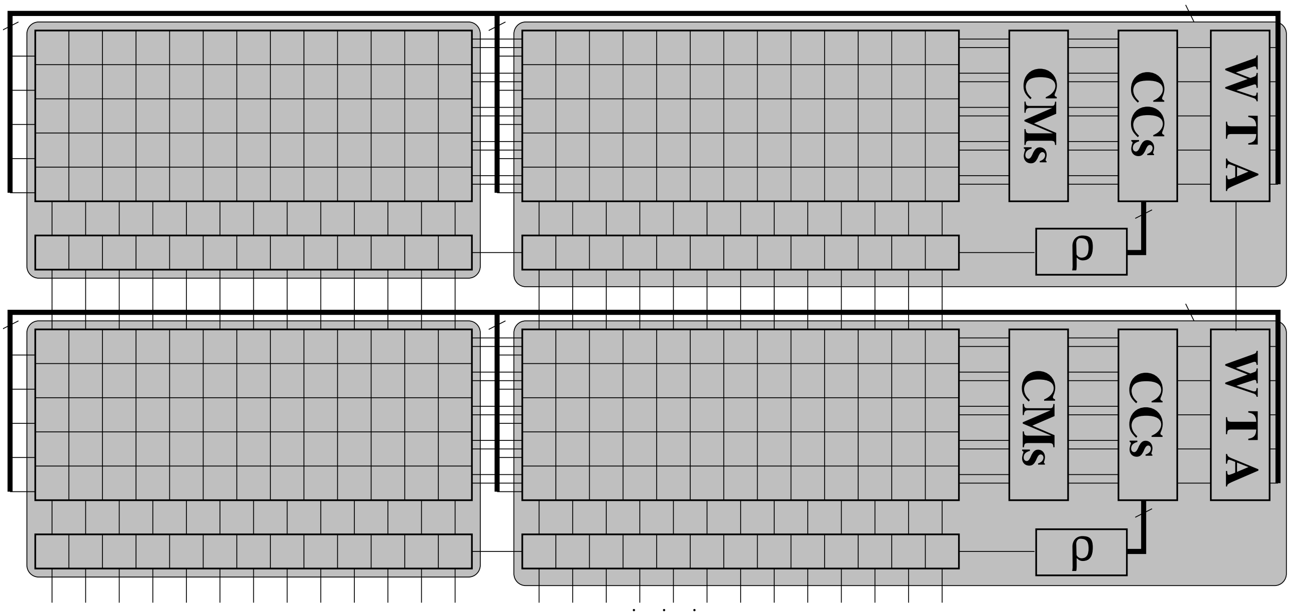


Fig. 8: Interchip Connectivity for Modular System Expansion

their area low. For previous stages the transistors can be made larger to contribute with a smaller σ_k, because they are less in number and will not contribute significantly to the total transistor area. For current sources L_A, a circuit similar to that shown in Fig. 7 is used. Current L_B in Fig. 7 (and similarly current L_A) is injected externally into the chip so that parameter $\alpha = L_A/L_B$ can be controlled.

F. Weights Read Out:

The switches sw_1 to sw_{100} of Fig. 7 were added to enable reading out the internally learned synaptic weights z_{ij}, and test the progress of the learning algorithm. These switches are all ON during normal operation of the system. However, for weights read-out, all except one will be OFF. The switch that is ON is selected by a decoder inside the chip, so that only column i of the synaptic array of Fig. 3 injects the current $z_{ij}L_B$ to nodes N_j. All nodes N_j can be isolated from current mirrors $CMAj$, and connected to output pads to sense the currents $z_{ij}L_B$, thus measuring the values of z_{ij}.

G. Modular System Expansibility:

The circuit of Fig. 3 can be expanded both horizontally, increasing the number of input patterns from 100 to $100 \times N$, and vertically increasing the number of possible categories from 18 to $18 \times M$. Fig. 8 shows schematically the interconnectivity between chips in the case of a 2×2 array.

Vertical expansion of the system is possible by making several chips share the input vector terminals $I_1, \ldots, I_{100}$, and node V_{COMMON} of the WTA (see Fig. 5). Thus, the only requirement is that V_{COMMON} be externally accessible. Horizontal expansion is directly possible by making all chips in the same row share their N_j, N'_j, and N'' nodes, and isolating all except one of them, from the current mirrors $CMA1, \ldots, CMA18$, $CMB1, \ldots, CMB18$, and the adjustable gain ρ-mirror. Also, all synapse inputs $\bar{y}_j$ must be shared.

Both vertical and horizontal expansion degrades the system performance. Vertical expansion causes degradation because the WTA becomes distributed among several chips. For the WTA of Fig. 5, all MA and MB transistors must match well, which is very unlikely if they are in different chips. A solution for this problem is to use a WTA topology based on current processing and replication, insensitive to inter-chip transistor mismatches [34], [35].

Horizontal expansion degrades the performance because current levels have to be changed:

- Either currents L_A and L_B are maintained the same, which makes the current mirrors $CMAj$, $CMBj$, CMM, $1{:}\rho$, CMC, the current comparators CCj, and the WTA to handle higher currents. This may cause malfunctioning due to eventual saturation in some of the blocks.
- Or currents L_A and L_B are scaled down so that the current mirrors $CMAj$, $CMBj$, CMM, $1{:}\rho$, CMC, the current comparators CCj, and the WTA handle the same current level. However, this produces an increase in mismatch between the current sources L_A and L_B.

IV. Experimental Results

A prototype chip that contains the previous circuit description of a real-time clustering engine has been fabricated in a standard double-poly double-metal 1.6μm CMOS digital process (Eurochip ES2). The die area is $1cm^2$ and it has been mounted in a 120-pin PGA package. This chip implements an ART1 system with 100 nodes in the *F1* layer and 18 nodes in the *F2* layer. Most of the pins are intended for test and characterization purposes. All the subcircuits in the chip can be isolated from the rest and conveniently characterized. The *F1* input vector **I**, which has 100 components, has to be loaded serially through one of the pins into a shift register. The time delay measurements reported in this paper do not include the time for loading the shift register.

The experimental measurements provided in this Section have been divided into four parts. The first describes DC characterization results of the elements that contribute critically to the overall system precision. These elements are the WTA circuit and the synaptic current sources. The second describes time delay measurements that contribute to the global throughput time of the system. The third presents system level experimental behaviors obtained with digital test equipment (HP82000). Finally, the fourth focuses on yield and fault tolerance characterizations.

A. System Precision Characterizations:

The ART1 chip was intended to achieve an equivalent 6-bit (~1.5% error) precision. The part of the system that is responsible for the overall precision is formed by the components that perform analog computations. These components are (see Fig. 3) all current sources L_A and L_B, all current mirrors $CMAj$, $CMBj$, CMM, CMC, and the ρ-mirror, the current comparators CCj, and the WTA circuit. The most critical of these components (in precision) is the WTA circuit. Current sources and current mirrors can be made to have mismatch errors below 0.2% [33], [36]-[38], at the expense of increasing transistors area and current, decreasing distances between matched devices, and using common centroid layout techniques [39]. This is feasible for current mirrors $CMAj$, $CMBj$, CMM, CMC, and the ρ-mirror, which appear in small numbers. However, the area and current level is limited for the synaptic current sources L_A and L_B, since there are many of them. Therefore, WTA and current sources L_A and L_B are the elements that limit the precision of the overall system, and their characterization results will be described next.

T_j	$10\mu A$	$100\mu A$	$1mA$
$\sigma(T_j)$	1.73%	0.86%	0.99%

Table 1. Precision of the WTA

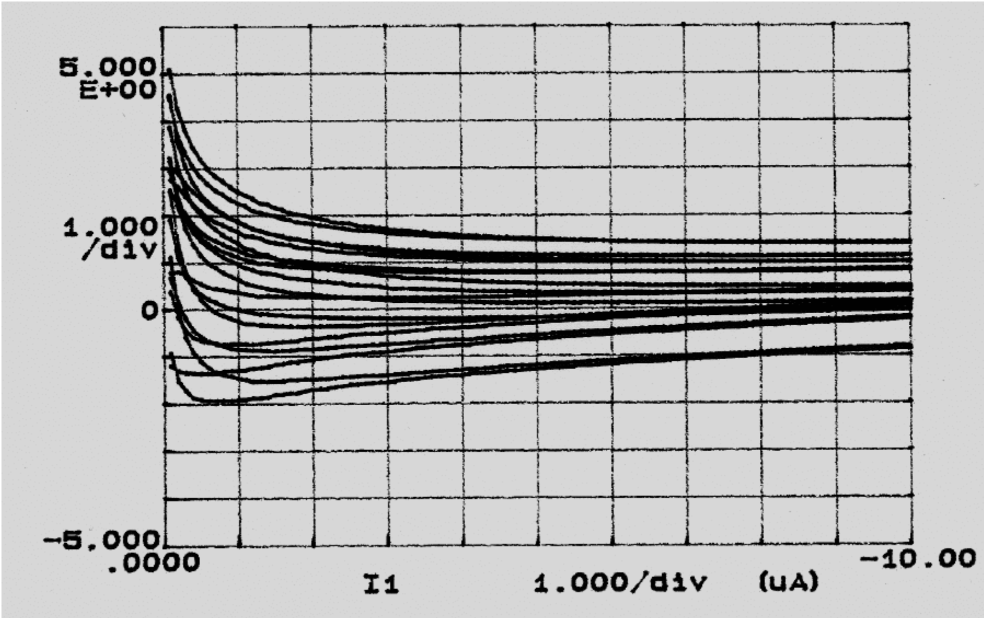


Fig. 9: Measured Mismatch error (in %) between 18 arbitrary L_A current sources

A.1 : WTA Precision Measurements:

L_A and L_B will have current values of 10μA or less. The maximum current a WTA input branch can receive is (see eq. (12)),

$$T_j\big|_{max} = L_M + \left[\sum_{i=1}^{100} z_{ij}(L_A I_i - L_B)\right]_{max} = L_M + 100\,(L_A - L_B) \qquad (16)$$

which corresponds to the case where all z_{ij} and I_i values are equal to '1' (remember that $L_A > L_B > 0$). In our circuit the WTA was designed to handle input currents of up to $1.5mA$ for each input branch. In order to measure the precision of the WTA, all input currents except two were set to zero. Of these two inputs one was set to $100\mu A$ and the other was swept between $98\mu A$ and $102\mu A$. This will cause their corresponding output voltages $\bar{y}_j$ to indicate an interchange of winners. The transitions do not occur exactly at $100\mu A$. Moreover, the transitions change with the input branches. The standard deviation of these transitions was measured as $\sigma=0.86\mu A$ (or 0.86%). Table 1 shows the standard deviation (in %) measured when the constant current is set to $10\mu A$, $100\mu A$, and $1mA$.

A.2 : Synaptic Current Sources Precision Measurements:

The second critical precision error source of the system is the mismatch between synaptic current sources. In our chip each of the 3700 L_A current sources and each of the 1800 L_B current sources could be isolated and independently characterized. Fig. 9 shows the measured mismatch error (in %) for 18 arbitrary L_A current sources when sweeping L_A between $0.1\mu A$ and $10\mu A$. As can be seen in Fig. 9, for currents higher than $5\mu A$

the standard deviation of the mismatch error is close to 1%. The same result is obtained for the L_B current sources.

B. Throughput Time Measurements:

For a real-time clustering device the throughput time is defined as the time needed for each input pattern to be processed. During this time the input pattern has to be classified into one of the pre-existing categories or assigned to a new one, and the pre-existing knowledge of the system has to be updated to incorporate the new information the input pattern carries. From a circuit point of view, this translates into the measurement of two delay times:

- The time needed by the WTA to select the maximum among all $\{\bar{c}_j T_j\}$.
- The time needed by the synaptic cells to change z_{ij} from its old value to $y_j I_i z_{ij}$.

T_1^a	T_1^b	T_2	$T_3, \ldots T_{18}$	t_{d1}	t_{d2}
$0\mu A$	$200\mu A$	$100\mu A$	0	$550ns$	$570ns$
$0\mu A$	$1mA$	$500\mu A$	0	$210ns$	$460ns$
$100\mu A$	$150\mu A$	$125\mu A$	$100\mu A$	$660ns$	$470ns$
$400\mu A$	$600\mu A$	$500\mu A$	$400\mu A$	$440ns$	$400ns$
$500\mu A$	$1.50mA$	$1.00mA$	$500\mu A$	$230ns$	$320ns$
$90\mu sA$	$110\mu A$	$100\mu A$	0	$1.12\mu s$	$1.11\mu s$
$490\mu A$	$510\mu A$	$500\mu A$	0	$1.19\mu s$	$1.06\mu s$
$990\mu A$	$1.01mA$	$1.00mA$	0	$380ns$	$920ns$

Table 2. Delay times of the WTA

B.1 : WTA Delay Measurements:

The delay introduced by the WTA depends on the current level present in the competing input branches. This current level will depend on the values chosen for L_A, L_B, and L_M, as well as on the input pattern $\mathbf{I}$ and all internal weights $\mathbf{z}_j$. To keep the presentation simple, delay times will be given as a function of T_j values directly. Table 2 shows the measured delay times when T_1 changes from T_1^a to T_1^b, and T_2 to T_{18} have the values given in the table. t_{d1} is the time needed by category y_1 to win when T_1 switches from T_1^a to T_1^b, and t_{d2} is the time spent by category y_2 in winning when T_1 decreases from T_1^b to T_1^a. As can be seen, this delay is always below $1.2\mu s$.

For the cases when the vigilance criterion is not directly satisfied and hence comparators CCj cut some of the T_j currents, an additional delay is observed. This extra delay has been measured to be less than $400ns$ for the worst cases. Therefore, the time needed until the WTA selects the maximum among all $\{\bar{c}_j T_j\}$ is less than $1.2\mu s + 0.4\mu s = 1.6\mu s$.

B.2 : Learning Time:

After a delay of $1.6\mu s$ (so that the WTA can settle), the learn signal $\overline{LEARN}$ (see Fig. 3) is enabled during a time t_{LEARN}. To measure the minimum t_{LEARN} time required, this time was set to a specific value during a training/learning trial, and it was checked that the weights had been updated properly. By

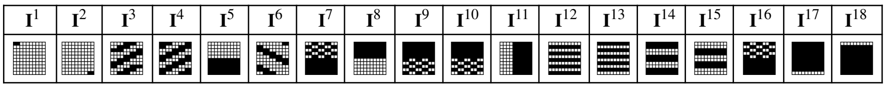


Fig. 10: Set of Input Patterns

progressively decreasing t_{LEARN} until some of the weights did not update correctly, it was found that the minimum t_{LEARN} time for proper operation was 190*ns*. By setting t_{LEARN} to 200*ns* and allowing the WTA a delay of $1.6\mu s$, the total throughput time of the ART1 chip is established as $1.8\mu s$.

B.3 : Comparison with Digital Neural Processors:

A digital chip with a feedforward speed of a connections per second, a learning speed of b connection updates per second, and a WTA section with a delay of c seconds must satisfy the following equation to achieve a throughput time of $1.8\mu s$ when emulating the ART1 algorithm of Fig. 2(c):

$$\frac{3700}{a} + \frac{100}{b} + c = 1.8\mu s \tag{17}$$

Note that there are 100 synapse weights z_{ij} to update for each pattern presentation, and 3700 feed-forward connections: 1800 connections to generate all $T_j = L_A|\mathbf{I} \cap \mathbf{z}_j| - L_B|\mathbf{z}_j| + L_M$, 1800 connections to generate $L_A|\mathbf{I} \cap \mathbf{z}_j|$, and 100 connections to generate $L_A|\mathbf{I}|$.

Assuming $c = 100ns$, and $a = b$, eq. (17) results in a processing speed of $a = b = 2.2 \times 10^9$ connections/s and connection-updates/s. A digital neural processor would require such figures of merit to equal the processing time of the analog ART1 chip presented in this paper. Therefore, this "approximate reasoning" makes us conclude that our chip has an equivalent computing power of $a + b = 4.4 \times 10^9$ connections/s plus connection-updates/s.

C. System Level Performance:

Although the internal processing of the chip is analog in nature, its input (I_i) and output ($\bar{y}_j$) are binary valued. Therefore, the system level behavior of the chip can be tested using conventional digital test equipment. In our case we used the HP82000 IC Evaluation System.

An arbitrary set of 100-bit input patterns $\{\mathbf{I}^k\}$ was chosen, shown in Fig. 10. A typical clustering sequence is shown in Fig. 11, for $\rho = 0.7$ and $\alpha = L_A/L_B = 1.05$. The first column indicates the input pattern $\mathbf{I}^k$ that is fed to the *F1* layer. The other 18 squares (10×10 pixels) in each row represent each of the internal $\mathbf{z}_j$ vectors after learning is finished. The vertical bars to the right of some $\mathbf{z}_j$ squares indicate that these categories won the WTA competition while satisfying the vigilance criterion. Therefore, such categories correspond to $\mathbf{z}_J$, and these are the only ones that are updated for that input pattern $\mathbf{I}^k$ presentation. The figure shows only two iterations of input patterns presentation, because no change in weights were observed after these. The last row of weights $\mathbf{z}_j$ indicates the resulting categorization of the input patterns. The numbers below each category indicate the input patterns that have been clustered into this category. In the following figures we will show only this last row of learned patterns together with the pattern numbers that have been clustered into each category.

Fig. 12 shows the categorizations that result when tuning the vigilance parameter ρ to different values while the currents were set to $L_A = 3.2\mu A$, $L_B = 3.0\mu A$, and $L_M = 400\mu A$ ($\alpha = L_A/L_B = 1.07$). Note

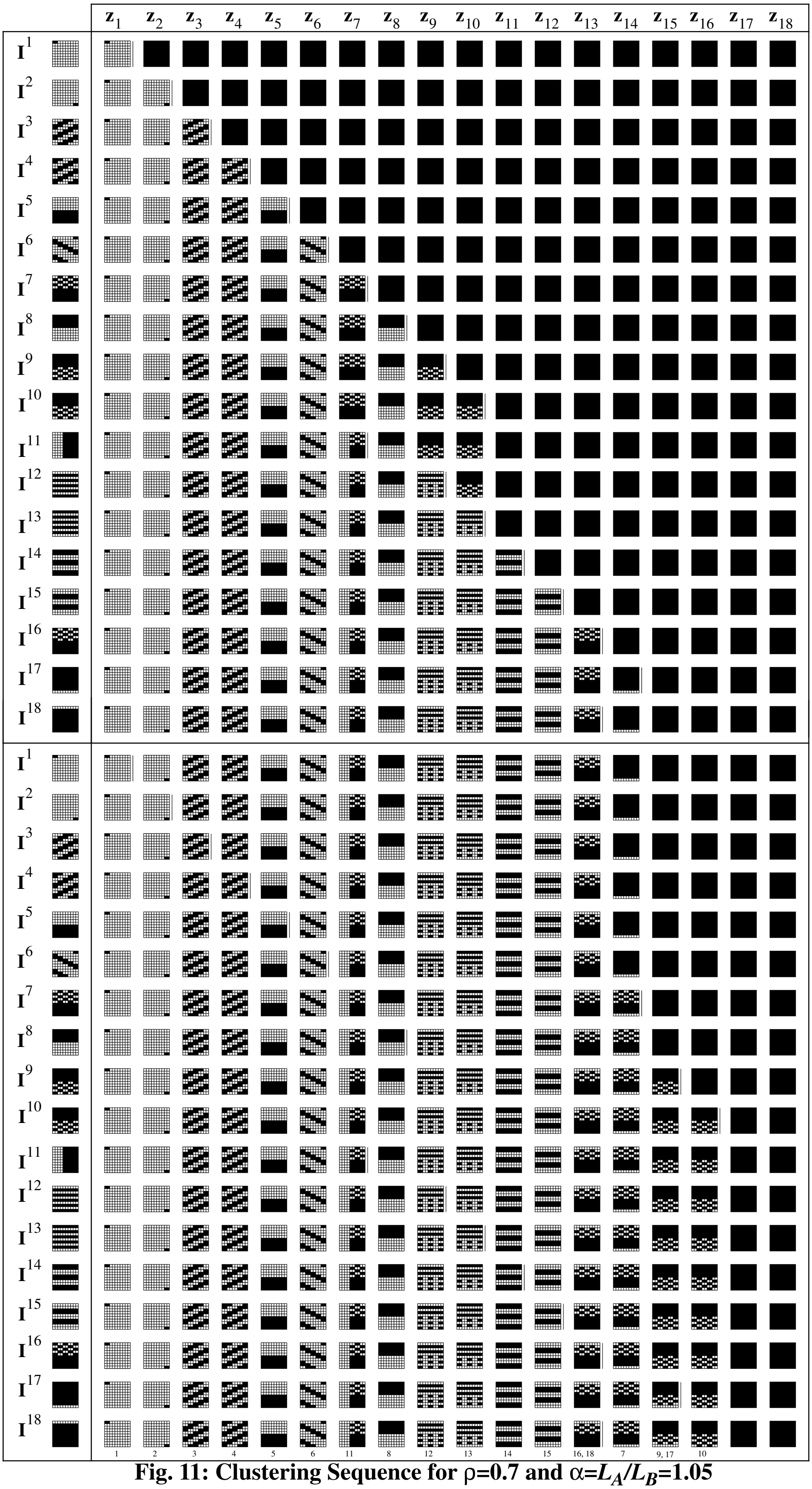


Fig. 11: Clustering Sequence for ρ=0.7 and α=L_A/L_B=1.05

	Catastrophic Faults (digital sense)								Non-Catastrophic Faults (digital sense)					
chip #	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**
1					x						x			
2		x					x		x	x		x	x	
3									x	x		x		x
4		x							x	x		x		
5	x	x							x					
6			x	x					x	x	x			
7			x						x					
8									x					
9		x		x					x	x				
10			x	x					x	x				
11	x													
12				x						x				
13	x	x	x						x	x				
14		x							x					
15		x	x						x	x				
16			x						x					
17	x		x						x					
18									x	x		x		
19										x				
20		x	x	x					x	x		x		
21	x								x					
22									x			x		
23	x								x					
24				x						x				
25														
26									x					
27	x	x	x						x					
28														
29	x													
30									x	x		x		

Table 3. Fault Characterizations of the 30 ART1 Chip Samples. Dark Shades: Sample with Catastrophic Fault; Light Shade: Sample with no Catastrophic Fault but with non Catastrophic Fault; no Shade: Sample with no Fault.

that below some categories there is no number. This is a known ART1 behavior: during the clustering process some categories might be created that will not represent any of the training patterns. In Fig. 13 the vigilance parameter is maintained constant at $\rho = 0$, while α changes from 1.07 to 50. For a more detailed explanation on how and why the clustering behavior depends on ρ and α see references [2] and [3], or other ART1 theoretical papers [1], [40].

D. Yield and Fault Tolerance:

A total of 30 chips (numbered 1 through 30 in Table 3 and Fig. 14) were fabricated. For each chip every subcircuit was independently tested and its proper operation verified; 14 different faults were identified. Table 3 indicates the faults detected for each of the 30 chips. The faults have been denoted from **F1** to **F14**, and are separated into two groups:

- *Catastrophic Faults (digital sense)* are those clearly originated by a short or open circuit failure. These faults are **F1**, ... **F8**. This kind of faults would produce a failure in a digital circuit.

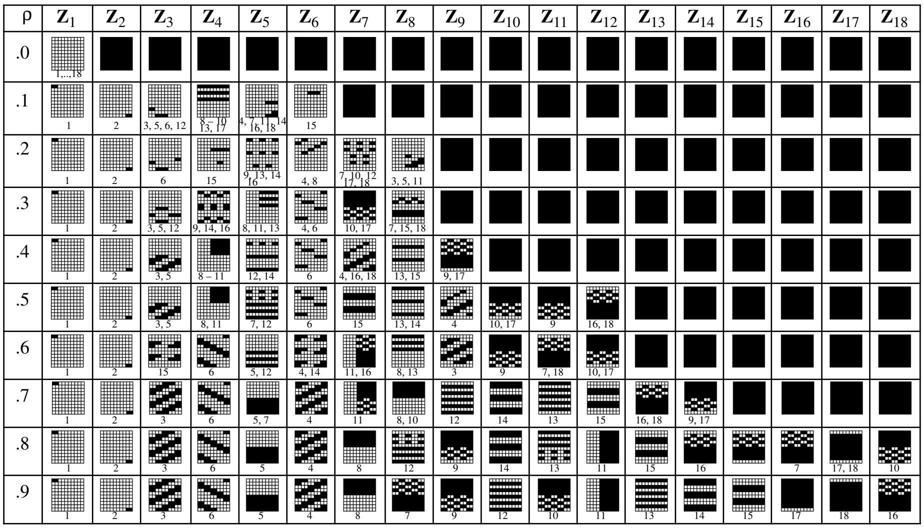


Fig. 12: Categorization of the input patterns for L_A=3.2µA, L_B=3.0µA, L_M=400µA, and different values of ρ

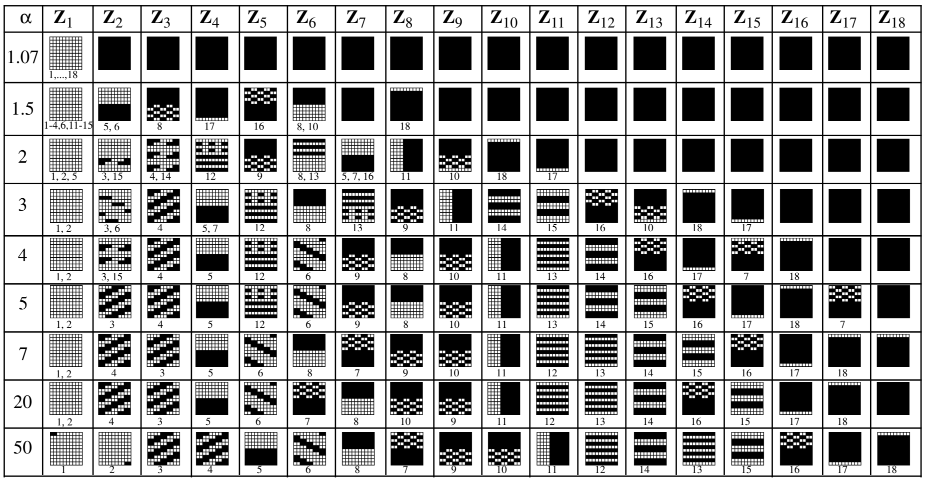


Fig. 13: Categorization of the input patterns for ρ=0 and different values of α

- *Non-Catastrophic Faults (digital sense)* are those that produce a large deviation from the nominal behavior, too large to be explained by random process parameter variations. These faults are **F9**, ... **F14**. This kind of faults would probably not produce a catastrophic failure in a digital circuit, but be responsible for significant delay times degradations.

Table 4 describes the subcircuits where the faults of Table 3 were found. Note that the most frequent faults are **F2/F9** and **F3/F10**, which are failures in some current sources L_A or L_B, and these current sources occupy a significant percentage of the total die area. Fault **F1** is a fault in the shift register that loads the input vector $\mathbf{I}^k$. Fault **F2** is a fault in the WTA circuit. Therefore, chips with an **F1** or **F2** fault could not be tested for system level operation. Faults **F3** and **F9** are faults detected in the same subcircuits of the chip, with **F3** being catastrophic and **F9** non-catastrophic. The same is valid for **F4** and **F10**, **F5** and **F11**, and so on until **F8** and **F14**.

Note that only 2 of the 30 chips (6.7%) are completely fault-free. According to the simplified expression for the yield performance as a function of die area Ω and process defects density ρ_D [41],

F1	non-operative shift register for loading $\mathbf{I}^k$
F2	non-operative WTA circuit
F3/F9	fault in a current source L_A
F4/F10	fault in a current source L_B
F5/F11	fault in vigilance parameter ρ current mirror
F6/F12	fault in current mirror CMM
F7/F13	fault in current mirrors $CMAj$ or $CMBj$
F8/F14	fault in current mirror CMC

Table 4. Description of Faults

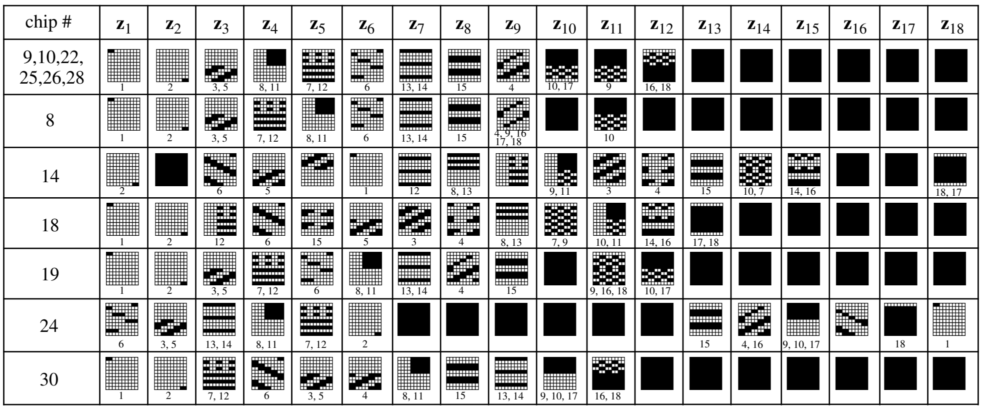


Fig. 14: Categorization of the input patterns performed by operative samples

$$yield = 100e^{-\rho_D\Omega} \tag{18}$$

this requires a process defect density[7] of $\rho_D = 3.2cm^{-1}$. On the other hand, ignoring the non-catastrophic faults yields 9 out of 30 chips (30%). According to eq. (18) such a yield would be predicted if the process defect density is $\rho'_D = 1.4cm^{-1}$.

Even though the yield is quite low, many of the faulty samples were still operative. This is due to the fault tolerant nature of the neural algorithms in general [42]-[45], and the ART1 algorithm in particular. Looking at Table 3 we can see that there are 16 chips that have an operative shift register and WTA circuit. We performed system level operation tests on these chips to verify if they would be able to form clusters of the input data, and verified that 12 of these 16 chips were able to do so. Moreover, 6 (among which were the two completely fault-free chips) behaved exactly identically. The resulting clustering behavior of these 12 chips is depicted in Fig. 14 for $\rho = 0.5$ and $\alpha = 1.07$.

V. Further Enhancements

The chip described in this paper is the first prototype designed by the authors for real-time clustering. As such, the design focused on testability and full characterization possibilities, instead of maximizing speed and yield, for example.

7. The effective die area is $\Omega = (0.92cm)^2$ to account for a $400\mu m$ width pad ring.

To make this chip an industry ready prototype, several trivial modifications should be introduced:

- First, substitute the serially loaded shift register that holds the input pattern $\mathbf{I}^k$, with some kind of parallel loading mechanism (using either electrical or optical data acquisition techniques).
- Use some simple yield enhancement technique. Looking at Table 3 and Table 4 we can see that most of the failures are due to faults in the synaptic current sources. A simple yield enhancement technique would be to add a number of spare columns of synapses, some of which would substitute faulty columns of synapses.
- Add a handshaking mechanism that would allow the chip to communicate with the outside circuitry. Thus, when the WTA produces a fast response (which, by the way, is most of the time), the outside circuitry need not wait for the worst case WTA delay.

Other, less trivial, enhancements that should be addressed relate to the high area and current consumption of the synaptic current sources L_A and L_B. One possibility would be to use UV-activated floating-gate-calibrated [46]-[49] current sources, instead of the tree-like structure of Fig. 7. In principle, it should be possible to use one single calibrated MOS transistor per synaptic current source. This transistor, which can be close to minimum size, does not have to drive a large current either. Calibration errors of 0.2% have been reported for currents of 200*nA* [49]. Using a scheme like this significantly reduces the current and silicon area consumption per synapse, allowing a much higher number of synapses per chip and thus boosting the performance of the chip significantly.

Other considerations relate to the question of how this chip would scale up with size. What would be the practical limitations? Usually a strong limitation when scaling up analog neural hardware is how systematic offsets accumulate. A common circuit technique for analog neural VLSI is the use of transconductors [22], [50]. Connecting many of them in parallel results in addition of their systematic offset components. If the size of the system is sufficiently large, this total offset can drive the system out of working range[8]. For our circuit the accumulation of systematic offsets of the synaptic current sources is not a problem. Note that the total currents T_j (which certainly include a common systematic offset) will compete in a WTA circuit, and the maximum among all $\{T_j\}$ is the same regardless of the presence or not of a common offset component.

A real scaling limitation for the circuit technique used in our chip is the following. The smallest current per synaptic current source is limited by the precision we want to achieve (even when using UV-activated floating-gate calibration techniques). Therefore, the maximum number of synapses that can be put into the same chip will be limited by the maximum power dissipation allowed by the package for a given precision. This implies a trade-off between precision and size.

Another problem that might arise when the number of nodes in the *F2* layer (maximum number of categories) becomes significantly large, is that the WTA circuit might not be able to detect the maximum among a large number of close-to-maximum inputs. At that point, one might reconsider if it is necessary to have an *F2* layer that provides one (and only one) winner, instead of an *F2* layer that provides a “bubble” of winners [51], [52].

A different way of system growth is to assemble different ART1 subsystems to perform supervised clustering tasks [53], or to combine ART cells hierarchically for higher level knowledge processing [54], [55].

---

8. In this case a global offset calibration technique can be used to overcome this problem.

VI. Conclusions

This paper presented the algorithm, circuit implementation, and experimental test results of an analog (digital-compatible) current-mode VLSI clustering engine. The algorithm is mainly based on the popular ART1 architecture, although slight modifications have been introduced to produce more efficient hardware. The presented prototype chip realizes an architecture with 100 *F1* nodes and 18 *F2* nodes, and is thus able to cluster 100-bit input patterns into up to 18 different categories. Modular expansibility is possible by directly assembling a matrix array of chips, without any extra interfacing circuitry. It has been shown that a digital neurocomputer able to provide the same throughput speed should have a processing speed of 2.2×10^9 connections per second and connections updates per second. Extensive chip characterization results have been given including system precision measurements, system speed measurements, system level clustering behavior, fault characterizations, and system level clustering behavior of faulty chip samples.

Finally, some improvement possibilities have been highlighted to enhance the efficiency and performance of the overall system for industrial production of commercial prototypes.

VII. References

[1] G. A. Carpenter and S. Grossberg, “A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine,” *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

[2] T. Serrano-Gotarredona and B. Linares-Barranco, “A VLSI-friendly ‘Fast-Learning’ ART1 Algorithm,” *Proceedings of the 1995 World Congress on Neural Networks,* Washington DC, vol. I, pp. 27-30, 1995.

[3] T. Serrano-Gotarredona and B. Linares-Barranco, “A Modified ART1 Algorithm more suitable for VLSI Implementations,” *Neural Networks*, accepted for publication.

[4] M. Griffin, G. Tahara, K. Knorpp, and W. Riley, “An 11-million Transistor Neural Network Execution Engine,” *Proc. of the 1991 IEEE Int. Conf. on Solid-State Circuits,* 1991, pp. 180-181.

[5] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, K. Shibata, M. Ooyama, M. Yamada, T. Sakaguchi, and M. Hashimoto, “A Self-Learning Neural Network Composed of 1152 Digital Neurons in Wafer-Scale LSIs,” *Proc. of the 1991 Int. Joint Conf. on Neural Networks,* Seattle, Washington, July 1991, pp. 1844-1849.

[6] N. Mauduit, M. Duranton, J. Gobert, and J. A. Sirat, “Lneuro 1.0: A Piece of Hardware LEGO for building Neural Network Systems,” *IEEE Trans. on Neural Networks,* vol. 3, No. 3, May 1992, pp. 414-422.

[7] A. J. De Groot and S. R. Parker, “Systolic Implementation of Neural Networks,” *SPIE High Speed Computing II*, vol. 1058, pp. 182-190, Los Angeles, California, 1989.

[8] S. Jones, K. Sammut, Ch. Nielsen, and J. Staunstrup, “Toroidal Neural Network: Architecture and Processor Granularity Issues,” in *VLSI Design of Neural Networks*, U. Ramacher and U. Rueckert (Eds.), pp. 229-254, Kluwer Academic Publishers, Dordrecht, Netherlands, 1991.

[9] R. W. Means and L. Lisenbee, “Floating-point SIMD Neurocomputer Array Processor,” *paper distributed by HNC Inc.,* 1993.

[10] U. Ramacher, J. Beichter, W. Raab, J. Anlauf, N. Bruels, U. Hachmann, and M. Wesseling, “Design of a 1st Generation Neurocomputer,” in *VLSI Design of Neural Networks*, U. Ramacher and U. Rueckert (Eds.), pp. 271-310, Kluwer Academic Publishers, Dordrecht, Netherlands, 1991.

[11] M. A. Viredaz, C. Lehmann, F. Blayo, and P. Ienne, “MANTRA: A Multi-Model Neural-Network Computer,” *Proc. of the 3rd Int. Workshop on VLSI for Neural Networks and Artificial Intelligence,* Oxford, September 1992.

[12] J. H. Chung, H. Yoon, and S. R. Maeng, “A Systolic Array Exploiting the Inherent Parallelisms of Artificial Neural Networks,” *Microprocessing and Microprogramming,* vol. 33, pp. 145-159, 1992.

[13] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed., Berlin, Germany: Springer-Verlag, 1989.

[14] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York: Plenum, 1981.

[15] R. Duda and P. Hart, *Pattern Classification and Scene Analysis,* New York: Wiley, 1973.

[16] J. Hartigan, *Clustering Algorithms*, New York: Wiley, 1975.

[17] R. Dubes and A. Jain, *Algorithms that Cluster Data*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[18] Y. H. Pao, *Adaptive Recognition and Neural Networks,* Reading, MA: Addison Wesley, 1989.

[19] A. Rodríguez-Vázquez, S. Espejo, R. Domínguez-Castro, J. L. Huertas, and E. Sánchez-Sinencio, “Current-Mode Techniques for the Implementation of Continuous- and Discrete-Time Cellular Neural Networks,” *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 40, No. 3, March 1993, pp. 132-146.

[20] A. Rodríguez-Vázquez and M. Delgado-Restituto, “Generation of Chaotic Signals using Current-Mode Techniques,” *Journal of Intelligent and Fuzzy Systems,* vol. 2, No. 1, pp. 15-37, 1994.

[21] H. Oh and F. M. A. Salam, “Analog CMOS Implementation of Neural Network for Adaptive Signal Processing,” *Proc. of the 1994 IEEE Int. Symp. on Circuits and Systems (ISCAS’94),* London, 1994, pp. 503-506.

[22] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J. L. Huertas, “A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing,” *IEEE Trans. on Neural Networks,* vol. 4, No. 3, May 1993, pp. 445-455.

[23] E. Keulen, S. Colak, H. Withagen, and H. Hegt, “Neural Network Hardware Performance Criteria,” *Proc. of the 1994 Int. Conf. on Neural Networks (ICNN’94)*, Orlando, 1994, pp. 1885-1888.

[24] M. Riedmiller, “Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms,” *Int. Journal of Computer Standards and Interfaces* (Special Issue on Neural Networks), (5), 1994.

[25] B. Kosko, “Adaptive Bidirectional Associative Memories,” *Applied Optics,* vol. 26, pp. 4947-4960, December, 1987.

[26] Y. F. Wand, J. B. Cruz, and J. H. Mulligan, “On Multiple Training for Bidirectional Associative Memory,” *IEEE Trans. on Neural Networks,* vol. 1, September 1990, pp. 275-276.

[27] C. S. Ho, J. J. Liou, M. Georgiopoulos, G. L. Heileman, and C. Christodoulou, “Analogue Circuit Design and Implementation of an Adaptive Resonance Theory (ART) Neural Network Architecture,” *Int. Journal on Electronics*, vol. 76, No. 2, pp. 271-291, 1994.

[28] S. W. Tsay and R. W. Newcomb, “VLSI Implementation of ART1 Memories,” *IEEE Transactions on Neural Networks*, vol. 2, No. 2, pp. 214-221, March 1991.

[29] D. C. Wunsch II, T. P. Caudell, C. D. Capps, R. J. Marks II, and R. A. Falk, “An Optoelectronic Implementation of the Adaptive Resonance Neural Network,” *IEEE Transactions on Neural Networks*, vol. 4, No. 4, pp. 673-684, July 1993.

[30] J. Lazzaro, R. Ryckebush, M. A. Mahowald, and C. Mead, “Winner-Take-All Networks of O(n) Complexity,” in *Advances in Neural Information Processing Systems,* vol. 1, D. S. Touretzky (Ed.), Los Altos, CA: Morgan Kaufmann, 1989, pp. 703-711.

[31] A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Medeiro and M. Delagdo-Restituto, “High Resolution CMOS Current Comparators: Design and Applications to Current-Mode Function Generation,” *Analog Integrated Circuits and Signal Processing*, Kluwer Academic Publishers, 7, pp. 149-165, 1995.

[32] T. Serrano and B. Linares-Barranco, “The Active-Input Regulated-Cascode Current Mirror,” *IEEE Trans. on Circuits and Systems-I: Fundamental Theory and Applications,* vol. 41, No. 6, June 1994, pp. 464-467.

[33] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, “Matching Properties of MOS Transistors,” *IEEE Journal of Solid-State Circuits,* vol. 24, October 1989, pp. 1433-1440.

[34] T. Serrano and B. Linares-Barranco, “A Modular Current-Mode High-Precision Winner-Take-All Circuit,” *Proc. of the 1994 Int. Symposium on Circuits and Systems,* London, 1994, vol. 5, pp. 557-560.

[35] T. Serrano and B. Linares-Barranco, “A Modular Current-Mode High-Precision Winner-Take-All Circuit,” *IEEE Trans. on Circuits and Systems II*, vol. 42, No. 2, pp. 132-134, February 1995.

[36] J. B. Shyu, G. C. Temes, and F. Krummenacher, “Random Error Effects in Matched MOS Capacitors and Current Sources,” *IEEE Journal Solid-State Circuits.,* vol. SC-19, No. 6, pp. 948-955, December 1984.

[37] K. R. Lakshmikumar, R. A. Hadaway, and M. A. Copeland, “Characterization and Modeling of Mismatch in MOS Transistors for Precision Analog Design,” *IEEE Journal Solid-State Circuits.,* vol. SC-21, No. 6, pp. 1057-1066, December 1986.

[38] C. Michael and M. Ismail, “Statistical Modeling of Device Mismatch for Analog MOS Integrated Circuits,” *IEEE Journal Solid-State Circuits.,* vol. 27, No. 2, pp. 154-166, February 1992.

[39] P. E. Allen and D. R. Holberg, *CMOS Analog Design*, Holt Rinehart and Winston Inc., New york, 1987.

[40] B. Moore, “ART1 and Pattern Clustering,” in *Proceedings of the 1988 Connectionist Summer School,* D. S. Touretzky, G. Hinton, and T. Sejnowski (Eds.), pp. 174-185, San Mateo, CA, 1989. Morgan Kaufmann.

[41] N. R. Strader and J. C. Harden, “Architectural Yield Optimization,” in *Wafer Scale Integration*, E. E. Swartzlander, Jr. (Ed.), pp. 57-118, Kluwer Academic Publishers, Boston, 1989.

[42] L. C. Chu, “Fault-tolerant Model of Neural Computing,” *IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pp. 122-125, 1991.

[43] C. Neti, M. H. Schneider, and E. D. Young, “Maximally Fault Tolerant Neural Networks,” *IEEE Trans. on Neural Networks,* vol. 3, No. 1, pp. 14-23, January 1992.

[44] J. H. Kim, C. Lursinsap, and S. Park, “Fault-Tolerant Artificial Neural Nertworks,” *Int. Joint Conf. on Neural Networks,* (IJCNN’91) Seattle, vol. 2, pp. 951, 1991.

[45] T. Petsche, “Trellis Codes, Receptive Fields, and Fault Tolerant, Self-Reparing Neural Networks,” in *Machine Learning: From Theory to Applications*, Cooperative Research at Siemens and MIT, S. J. Hanson, W. Remmele,

and R. L. Rivest (Eds.), Springer-Verlag, Berlin, Germany, pp. 241-268, 1993.

[46] D. A. Kerns, *Experiments in Very Large-Scale Analog Computations,* PhD Thesis, California Institute of Technology, Pasadena, California, 1993.

[47] G. Cauwenbergs, C. F. Neugebauer, and A. Yariv, “Analysis and Verification of an Analog VLSI Incremental Outer-Product Learning System,” *IEEE Trans. on Neural Networks*, vol. 3, No. 3, pp. 488-497, May 1992.

[48] K. Yang and A. G. Andreou, “The Multiple Input Floating Gate MOS Differential Amplifier: An Analog Computational Building Block,” *Proc. of the 1994 Int. Symp. on Circuits and Systems (ISCAS’94)*, London, pp. 37-40, 1994.

[49] H. Miwa, K. Yang, P. O. Pouliquen, N. Kumar, and A. G. Andreou, “Storage Enhancement Techniques for Digital Memory Based, Analog Computation Engines,” *Proc. of the 1994 Int. Symp. on Circuits and Systems (ISCAS’94)*, London, pp. 45-48, 1994.

[50] B. W. Lee and B. J. Sheu, *Hardware Annealing in Analog VLSI Neurocomputing*, Boston: Kluwer Academic Publishers, 1991.

[51] S. Grossberg, “Nonlinear Neural Networks: Principles, Mechanisms, and Architectures,” *Neural Networks*, vol. 1, pp. 17-61, 1988.

[52] T. Kohonen, *Self-Organization and Associative Memory*, Springer Verlag, New York, 1984.

[53] G. A. Carpenter, S. Grossberg, and J. H. Reynolds, “ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network,” *Neural Networks,* vol. 4, pp. 565-588, 1991.

[54] S. Grossberg, *Neural Networks and Natural Intelligence*, MIT Press, 1989.

[55] G. A. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, 1991.