A VLSI Architecture for Fast Clustering with Fuzzy ART Neural Networks

E. Granger†, Y. Blaquière‡, Y. Savaria†, M.-A. Cantin‡, and P. Lavoie§ *†

† Electrical and Computer Engineering Dept., École Polytechnique de Montréal, P.O. Box 6079, station A, Montreal, Qc., Canada, H3C 3A7.

‡ Computer Science Dept., Université du Québec à Montréal, P.O. Box 8888, station A, Montreal, Qc., Canada, H3C 3P8.

§ Defence Research Establishment Ottawa, Department of National Defence, Canada, K1A 0Z4.

Abstract

The hardware implementation of the Fuzzy ART neural network applied to a demanding real time radar signal clustering problem is investigated. To obtain efficient solutions for implementing this neural network with dedicated hardware, the network's algorithm is reformulated, and then a novel Fuzzy ART system architecture is proposed. This system architecture is composed of a global comparator and several identical elementary modules (EMs), each one emulating a number of neurons. The general architecture of each EM consists of a local comparator, dividers, neural processors, and a block of memory.

1: Introduction

On-line clustering of intercepted radar pulses is important for data reduction in electronic support measures (ESM) systems. The purpose of radar ESM is to search for, intercept, locate, and analyze radar signals in the context of military surveillance. It is an important preliminary step in all electronic warfare systems, which allows the evaluation of the countermeasures to be undertaken for self-protection [1]. The need for data reduction in radar ESM arises from the signal densities encountered in certain radar bands, which can reach up to 10^6 radar pulses per second. Grouping radar pulses into categories corresponding to active radar emitters at the early stages of processing could considerably reduce the processing requirements, as well as hardware cost of ESM systems [2] [3] [4]. Since the number and description of the radar emitters

*The authors would like to acknowledge the contribution of Eric Audet.‡

†This work was supported in part by the Defence Research Establishment Ottawa, the Canadian Microelectronics Corporation, le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche, and the Natural Sciences and Engineering Research Council of Canada.

are *a priori* unknown, the radar pulses must be grouped into categories based on their perceived similarity, a process which is called clustering [5].

The Fuzzy ART neural network [6] is capable of fast, stable, and unsupervised learning of categories in response to non-stationary sources of binary or analog patterns. This network is of interest for our application, since it can cluster patterns autonomously in real-time, without prior knowledge of the number of categories. This paper presents a study of Fuzzy ART neural network implementation using dedicated VLSI hardware applicable to data reduction in ESM systems. The main features of the Fuzzy ART neural network are briefly summarized in the next section. In Section 3, this network's algorithm is reformulated for high speed clustering problems. Finally, in Section 4, a VLSI architecture is proposed to implement this algorithm for ESM systems and other high speed clustering applications.

2: Fuzzy ART neural network

2.1: Neural network model

Adaptive Resonance Theory (ART) was introduced by Carpenter and Grossberg [7]. ART neural networks can develop stable recognition capability on-line by self-organization, in response to arbitrary sequences of input patterns. The Fuzzy ART neural network proposed by Carpenter, Grossberg and Rosen [6] can accept binary and analog inputs. It is essentially an adaptive, unsupervised learning network consisting of two layers of neurons that are fully connected: a $2M$ neuron input or *comparison* layer (F1) and a N neuron output or *competitive* layer (F2). A weight value w_{ji}, represented as a real in the interval [0,1], is associated with each connection. The indices i and j denote the neurons that belong to the layers F1 and F2 respectively. The set of weights $\vec{W} = \{w_{ji} : i = 1, 2, ..., 2M, j = 1, 2, ..., N\}$ encodes information that defines the categories learned by the network. These can be modified dynamically during network operation. For each neuron j of F2, the vector of adaptive weights $\vec{w}_j = (w_{j1}, w_{j2}, ..., w_{j2M})$ corresponds to the subset of weights ($\vec{w}_j \subset \vec{W}$) connected to neuron j. This vector $\vec{w}_j$ is named *prototype vector*, and it represents the set of characteristics defining the category j.

2.2: Algorithmic description of Fuzzy ART

The Fuzzy ART network's functionality can be described as an algorithm [8] (as shown in Fig. 1). This algorithm can be divided into five execution steps:

1. Weights and parameters initialization: Initially, all the neurons of F2 are uncommitted, and all weight values are initialized to 1. An F2 neuron becomes committed when it is selected for an input $\vec{a}$. Then the corresponding weights w_{ji} can take real values in the interval [0,1].

2. Input vector coding: When a new input vector $\vec{a} = (a_1, a_2, ..., a_M)$ of M elements (where each element a_i is a real number in the interval [0,1]) is presented to the network, it undergoes a preliminary coding. *Complement coding* of $\vec{a}$ results in a network input vector $\vec{I}$ of $2M$ elements such that: $\vec{I} = (\vec{a}; \vec{a}^c) = (a_1, a_2, ..., a_M; a_1^c, a_2^c, ..., a_M^c)$, with $a_i^c = 1 - a_i$. This coding is recommended [6] to prevent a category proliferation problem that may occur in analog ART networks [8].

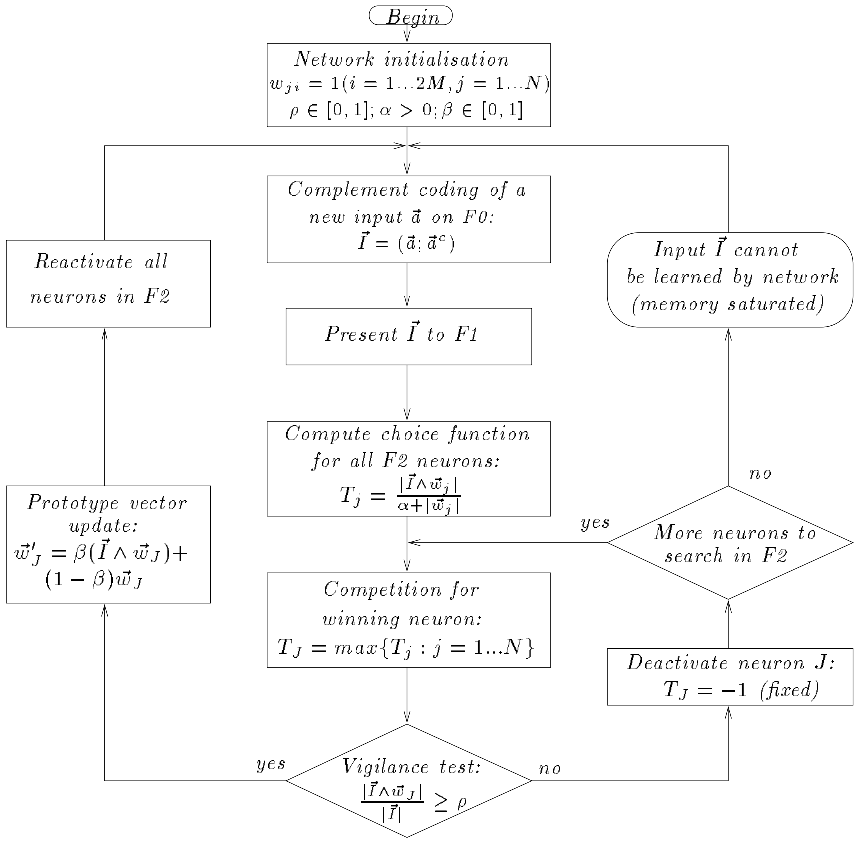

Figure 1. Flowchart representation of the Fuzzy ART algorithm.

3. Category choice: With each presentation of an input $\vec{I}$ to F1, the *choice function* $T_j(\vec{I})$ is calculated for each neuron j in F2:

$$T_j(\vec{I}) = \frac{|\vec{I} \wedge \vec{w}_j|}{\alpha + |\vec{w}_j|} \tag{1}$$

where $\wedge$ is the fuzzy logic AND operator ($\vec{I} \wedge \vec{w}j = [min(I_1, w_{j,1}), ..., min(I_{2M}, w_{j,2M})]$), $|\cdot|$ is the norm operator ($|\vec{x}| = \sum_{i=1}^{2M} |x_i|$), and α is a user-defined *choice parameter* such that $\alpha > 0$. F2 is a *winner-take-all* competitive layer, where the winner is the neuron $j = J$ with the greatest value of activation T_j for the input $\vec{I}$ ($T_J = max\{T_j : j = 1...N\}$). If the same T_J value is obtained by two or more neurons, the smallest j wins.

4. Vigilance test: This step serves to compare the similarity between the prototype vector of the winning neuron $\vec{w}_J$ and input $\vec{I}$ against a user-defined *vigilance parameter* ρ through the following test:

$$\frac{|\vec{I} \wedge \vec{w}_J|}{|\vec{I}|} \geq \rho \tag{2}$$

where $\rho = [0, 1]$. This comparison is carried out on layer F1: the winning neuron J transmits its $\vec{w}_J$ to F1 for comparison with $\vec{I}$. If the vigilance test (Eq. 2) is passed, the neuron J becomes selected, and it is allowed to adapt its prototype vector (Step 5). Otherwise, the neuron J is deactivated for the current input $\vec{I}$: T_J is set equal to -1 for the duration of the current input presentation. The algorithm searches through the remaining F2 layer neurons (Steps 3 and 4), until some other neuron J passes the vigilance test. If no committed neuron from the F2 layer can pass this test, an uncommitted neuron is selected.

5. Prototype vector update: The prototype vector of the winning neuron J is updated according to:

$$\vec{w}'_J = \beta(\vec{I} \wedge \vec{w}_J) + (1-\beta)\vec{w}_J \tag{3}$$

where β is a user-defined *learning rate parameter* such that $\beta = [0, 1]$. The algorithm can be set to slow learning, with $0 < \beta < 1$, or to fast learning, with $\beta = 1$. Once this update step is accomplished, the network can process a new input vector. The flowchart representation in Figure 1 summarizes the basic steps of the Fuzzy ART algorithm. It is assumed that the number of neurons N in layer F2 is fixed.

3: Reformulated Fuzzy ART algorithm

Prior to implementation in dedicated hardware, the Fuzzy ART algorithm may be reformulated. This circumvents the massively parallel processing requirements of the algorithm given in Fig. 1, and exploits the potential for re-use of data to maximize throughput, while minimizing communication costs and processing load. Computation of the choice functions (T_j for every committed neuron j), and their components ($|\vec{I} \wedge \vec{w}_j|$ and $|\vec{w}_j|$), constitutes a substantial proportion of the total processing effort. Fortunately, once the elements $|\vec{I} \wedge \vec{w}_j|$ and $|\vec{w}_j|$ have been computed, their values can be re-used on many occasions during the processing of an input. As for the choice functions, there is no need to compute the T_j value of a neuron j that would not pass the vigilance test. This test should thus occur as soon as the component $|\vec{I} \wedge \vec{w}_j|$ is available for a neuron j. Since complement coding is an input normalization such that, by definition, $|\vec{I}| = |(\vec{a}; \vec{a}^c)| = M$ [6], this test reduces to $|\vec{I} \wedge \vec{w}_j| \geq \rho \cdot M$. In parallel with the vigilance test, it is convenient to carry out a *subset test* in order to detect the situation where a prototype vector $\vec{w}_j$ is a subset of $\vec{I}$, that is, when $|\vec{I} \wedge \vec{w}_j| = |\vec{w}_j|$. If a neuron j is a subset choice of $\vec{I}$, then S_j is set equal to 1. If this neuron j is chosen and it passes the vigilance test, then its prototype vector $\vec{w}_J$ remains unchanged during the learning phase, since $\vec{w'}_J = \vec{I} \wedge \vec{w}_J = \vec{w}_J$. Therefore, the learning phase can be bypassed, and the network speed increased.

These observations lead to an alternate Fuzzy ART algorithm representation, which is shown in Fig. 2 for the fast learning case ($\beta = 1$). The algorithm sequentially processes committed neurons only, and N now represents the number of committed neurons, which starts from 1 and increases progressively as learning takes place. The reformulated algorithm is divided into two parts: a *recall phase* and a *learning phase*. During the recall phase, the values T_j and S_j are computed for all committed neurons. At the end, the winning neuron J, is chosen. During the learning phase, in the case where $T_J \neq 0$, the winning neuron J corresponds to a committed category that passes the vigilance test. If J is a subset choice for $\vec{I}$ ($S_J = 1$), then the prototype vector update is bypassed; otherwise ($S_J \neq 1$), the prototype vector $\vec{w}_J$ is updated according to Eq. 3. In the case where $T_J = 0$, no committed category has passed

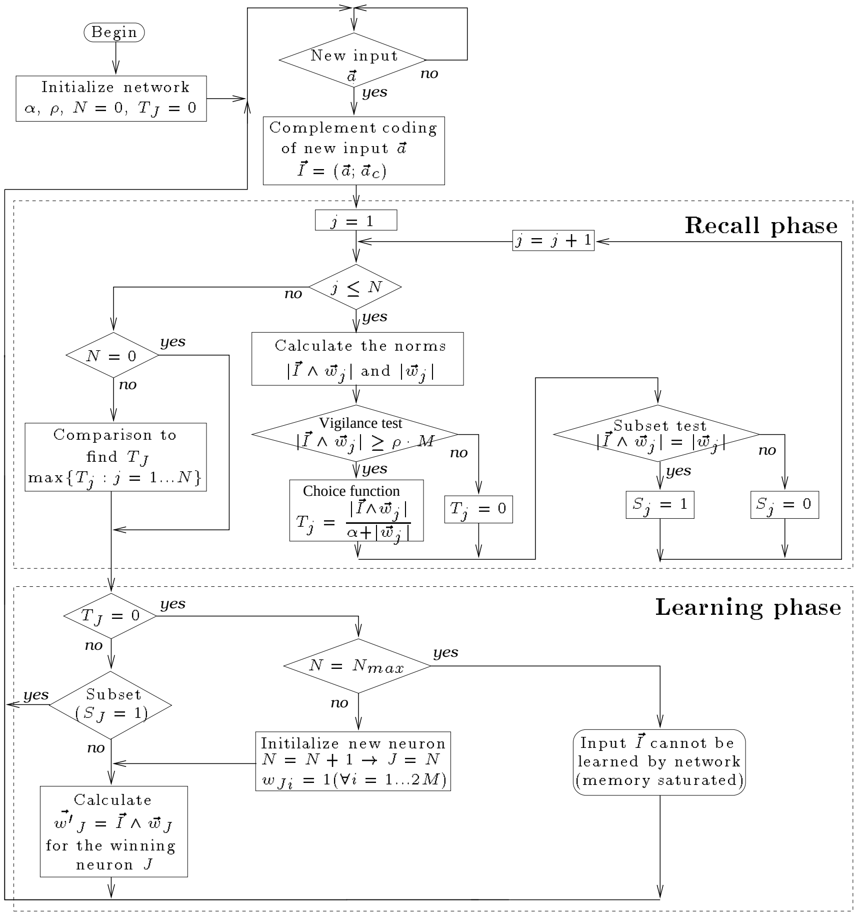

Figure 2. Alternate representation of the Fuzzy ART algorithm.

the vigilance test. If the maximum number of neurons is not attained ($N < N_{max}$), then an uncommitted neuron $J = N + 1$ is assigned to $\vec{I}$; otherwise ($N = N_{max}$), the network is unable to categorize $\vec{I}$ because its memory is saturated. After the learning phase, the network is ready to accept another input vector.

4: Fuzzy ART system architecture

In this section, we present a VLSI architecture for the reformulated Fuzzy ART algorithm of Fig. 2. Bearing in mind ESM applications, we sought a Fuzzy ART network with the following parameters: $2M = 32$ neurons in the F1 layer, $N_{tot} \cong 200$ neurons (categories) in the F2 layer, fast learning ($\beta = 1$), and $b = 11$ bit word length. Computer simulations against radar data with b equal to 11 bits produces errors within 2% of the results obtained with floating-point values. For proof-of-concept, the Fuzzy ART network should accept and categorize a new input vector every 10μs. Given the constraints imposed by current VLSI technology, a dedicated VLSI hardware implementation of the processing must be distributed over several integrated circuits. A detailed processing rate estimation (not reported here for brevity) has been performed. This analysis showed that the proposed system can meet these performance requirements.

The proposed Fuzzy ART system architecture (Fig. 3) is composed of a global

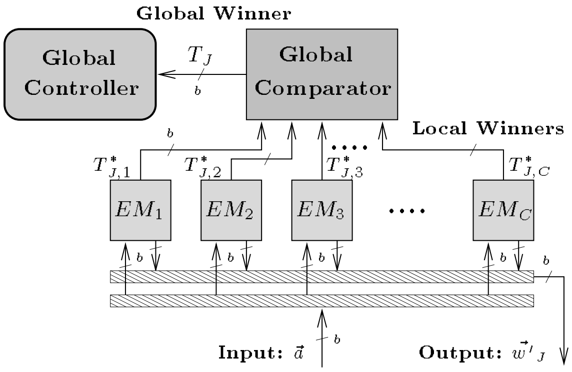

Figure 3. Architecture of Fuzzy ART system.

comparator, a global controller, and C identical elementary modules (EMs), each of which emulates $N = \frac{N_{tot}}{C}$ neurons. Each EM determines the T^*_J of its local winner J^*, for an input $\vec{I}$, considering the N neurons it emulates. The global comparator then selects the global winner. The general data path architecture of an EM (shown in Fig. 4) consists essentially of a local controller, a local comparator, D fixed-point dividers, P neural processors (NPs) - each emulating $\frac{N}{P}$ neurons -, $\frac{P}{D}$ NPs per divider, and random access memory (RAM) for storage of the prototype vectors. Fig. 5 presents the basic structure of a NP module. In agreement with the reformulated algorithm, a NP can be in the recall phase or the learning phase. During the recall phase, the value $|\vec{w}_j \wedge \vec{I}|$ is calculated for each committed F2 neuron j, and during the learning phase the prototype vector $\vec{w'}_J$ (and $|\vec{w'}_J|$) of the globally winning neuron is updated. To reduce the control burden, an EM is considered to be committed if any one of the N neurons it emulates is committed. That is, when needed, a new EM becomes activated, and computes its local T_J based on the results from all its internal neurons (even though these neurons may not all be formally committed). Therefore, the system architecture performs the recall and learning phases of the reformulated Fuzzy ART algorithm sequentially using the same hardware. The system's behavior is described for both of these phases in the following subsections.

The system speed depends mostly on the number of neurons, and on access time to the prototype vectors [12]. Storage of the prototype vectors for $N_{tot} \cong 200$ neurons, each of which consists of $2M = 32$ elements of $b = 11$ bits, would require about $70k$bits of RAM. Considering that memory is only part of the system, for a proof-of-concept implementation with the desired number of neurons, it would be preferable to use a cascade of several identical chips, with their own internal RAM. If there are C EMs, then each EM stores the weights of $N = \frac{N_{tot}}{C}$ neurons; and if there are P NPs per EM, then the memory associated with each NP contains the weights of $\frac{N}{P}$ neurons. On-circuit storage permits very fast processing distributed over C identical EMs with moderate I/O bandwidth requirements.

4.1: Recall phase

Complement coding is done inside each EM circuit to reduce I/O requirements. This coding has little impact on the system throughput, since the full complement coding of an M dimensional input overlaps the NPs processing. As an input $\vec{a}$ is being encoded inside an EM, the recall phase forms a data processing pipeline starting from

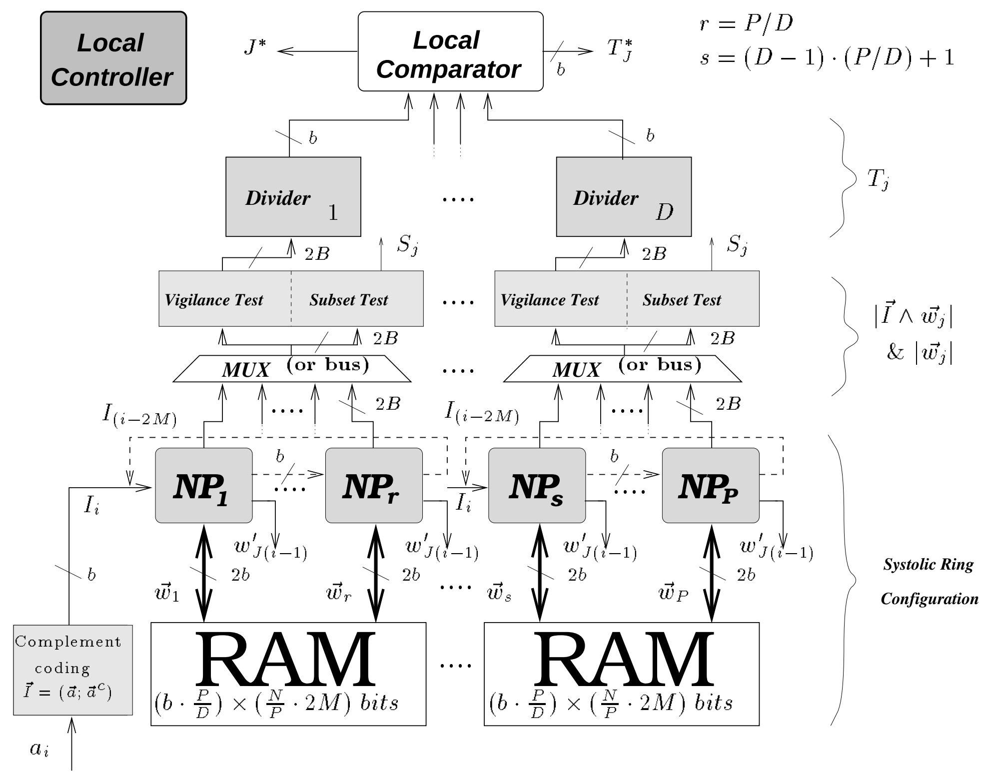

Figure 4. Architecture of an elementary module (EM).

the systolic ring NPs, and ending with the global comparators.

A *systolic ring configuration* [9] [11] is proposed for the NPs, to exploit the repetitive and regular nature of the $|\vec{I} \wedge \vec{w}_j|$ computation, and to increase the possibility for parallel execution with VLSI technology. This configuration is based on pipelined processing through a locally connected chain of NPs, communicating through unidirectional links (see Fig. 4). The shifting of data between NPs offers a larger total inter-NP throughput by using local communications only, yielding a better balance between communications and computations [10]. Furthermore, such regular communications minimize I/O requirements, allowing for a large number of NPs per EM. Each NP communicates with a local memory that contains the prototype vectors of $\frac{N}{P}$ neurons in the F2 layer. The $2M$ elements of the input vector $\vec{I}$ are shifted right in natural order sequence (I_1, I_2, etc.) through the NPs associated with a divider, while the $2M$ elements of every prototype vector are sequentially read from RAM and shifted upwards. Respective input vectors and weights for each neuron j are combined in a series of $2M$ minimum operations, $I_i \wedge w_{ji}$, so that each NP can accumulate an output value $|\vec{I} \wedge \vec{w}_j|$. Notice that for each input $\vec{I}$, $\frac{N}{P}$ cycles around each ring are required to process the entire set of emulated neurons. Fig. 5 shows that the value $|\vec{w}_j|$ (computed during the learning phase) is also generated from the NPs in the same sequence as its respective $|\vec{w}_j \wedge \vec{I}|$.

The values $|\vec{w}_j \wedge \vec{I}|$ and $|\vec{w}_j|$ are then transferred from each NP to its respective divider, after passing through vigilance and subset tests (carried out in parallel). If the value $|\vec{w}_j \wedge \vec{I}|$ of a neuron j passes the vigilance test, then this value is passed on to the divider; otherwise, T_j is set to 0, eliminating j from the competition. The

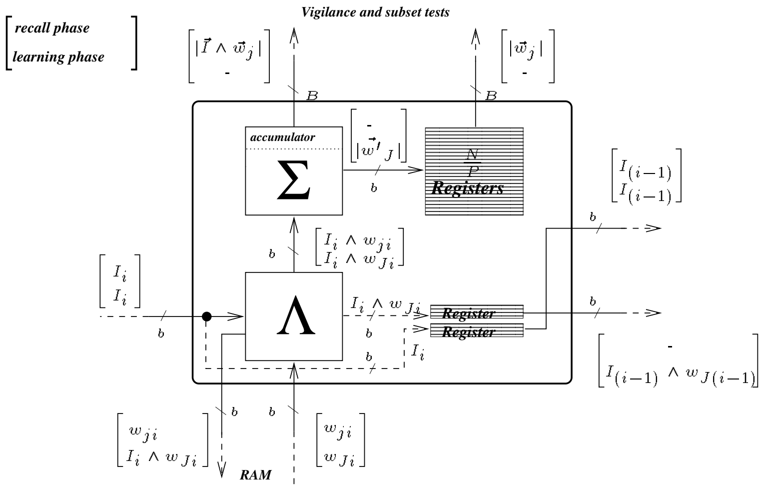

Figure 5. The basic structure of a NP module consists of a "min" comparator ($\wedge$), an "adder-accumulator" ($\sum$), and registers to compute the values $|\vec{I} \wedge \vec{w}_j|$, $\vec{w'}_J$ and $|\vec{w'}_j|$. Note the dual labels for recall (top) and learning (bottom) phases respectively.

corresponding value S_j is retained for use during the learning phase. Fixed-point dividers are used to compute the choice functions from the $|\vec{w}_j \wedge \vec{I}|$ and $(|\vec{w}_j| + \alpha)$ terms. Note that the value $(|\vec{w}_j| + \alpha)$ is computed as a preliminary step to its use by a divider. The division efficiency is very important to the system's performance: it must be reasonably fast, without sacrificing the accuracy of the results. To maximize performance, each divider contains pipeline stages to reduce the clock period.

Following the divisions, the resulting T_js are fed to the local comparator. The maximum T_j in the current cycle is compared to the maximum T_js of the previous cycles, until all the T_js have been compared, and a *local winner* J^* has been found. The index J^* of the local winner is then passed on to the local controller in anticipation of an update phase. The local T^*_J of each EM is then sent for global comparison to obtain a *global winner* J. These two "max" comparisons can be implemented using binary trees of comparators, with a processing time that grows as $O(log_2(D))$ and $O(log_2(C))$. The EM containing the global winner J is then activated for the learning phase.

4.2: Learning phase

During the learning phase, the prototype vector $\vec{w}_J$ of the winning neuron (with $S_J = 0$) is updated by its respective NP: the weights are loaded from memory, processed in the NP, then written back to memory. The processing of the learning phase is similar to that of the recall phase. Only the winning neuron's prototype elements are shifted through its NP, in sequence with $\vec{I}$, to compute $\vec{w'}_J = \vec{w}_J \wedge \vec{I}$. The other NPs linked to the same divider simply shift the input through. At the same time, $|\vec{w'}_J|$ is computed, and the elements w'_{Ji} are shifted outwards from its NP (as shown in Fig. 5). This allows external monitoring of new and updated categories in real time. Rather than recomputing the $|\vec{w}_j|$ for all the neurons in use at every recall phase, only the $|\vec{w}_J|$ of an updated neuron is recomputed, using the existing NP hardware ("adder-accumulator"). This $|\vec{w}_J|$ updating operation is done during the *learning phase*, while updating the actual $\vec{w}_J$. Therefore, once a winning neuron

J is chosen (with $S_J = 0$), the prototype $\vec{w}_J$, and its norm $|\vec{w}_J|$ are updated.

5: Conclusion

In this paper, we reformulated the Fuzzy ART algorithm to simplify its implementation, and proposed a VLSI system architecture that allows partitioning into several identical integrated circuits. This dedicated VLSI architecture is suitable for fast clustering of radar signals in ESM systems, and other high speed clustering applications. The validity of this approach has been demonstrated through processing rate estimation. Given the system's nature and current VLSI technology, it appears reasonable to expect a processing rate exceeding 10^5 patterns per second. A proof-of-concept CMOS implementation is currently under way at École Polytechnique de Montréal. A design has been described in VHDL, and is being targeted to BiCMOS 0.8μm technology, using Synopsys simulation and synthesis tools. This VLSI architecture represents just one among many possible architectures for the Fuzzy ART algorithm. The results of our current implementation, and future exploration should uncover an even broader range of architectural solutions.

References

[1] P. M. Grant, and J. H. Collins, *Introduction to Electronic Warfare*, Proc. of IEE (UK), vol. 129, Part. F. no. 3, pp. 113-129, June 1982.

[2] J. F. Crespo, P. Lavoie, and Y. Savaria, *Fast Convergence with Low Precision Weights in ART1 Networks*, Proc. of 1994 IEEE Int'l Symp. Circuits and Systems, London, England, vol. 6, pp. 237-204, 1994.

[3] B. Kamgar-Parsi, B. Kamgar-Parsi, and J. C. Sciortino Jr., *Automatic Data Sorting Using Neural Network Techniques*, Naval Research Laboratory Report NRL/FR/5720-96-9803, February 1996.

[4] J. A. Anderson, M. T. Gately, P. A. Penz, and D. R. Collins, *Radar Signal Categorization Using a Neural Network*, Proc. of the IEEE, vol. 78, no. 10, pp. 1646-1656, October 1990.

[5] M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, 1973.

[6] G. A. Carpenter, S. Grossberg,and D. B. Rosen, *Fuzzy ART: Fast Stable Learning and Categorisation of Analog Patterns by an Adaptive Resonance System*, Neural Networks, vol. 4, pp. 759-771, 1991.

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

[8] B. Moore, *ART1 and Pattern Clustering*, In Proc. of 1988 Connectionist Models Summer School, Eds. Touretski, Hinton and Sejnowski, Morgan Kaufmann Publishers, pp. 174-185, 1988.

[9] S. R. Jones, K. M. Sammut, C. H. Neilson, and J. Strauntrup, *Toroidal Neural Networks: Architectures and Processor Granularity Issues*, In VLSI Design of Neural Networks, Eds. Ramacher and Ruckert, pp. 229-254, 1991.

[10] S. Y. Kung, *Digital Neural Networks*, Prentice-Hall, pp. 337-406, 1993.

[11] S. R. Jones, K. M. Sammut, and J. Hunter, *Learning in Linear Systolic Neural Network Engines: Analysis ans Implementation*, IEEE Transactions on Neural Networks, vol. 5, no.4, pp. 584-593, July 1994.

[12] U. Ramacher, *Guide Lines to VLSI Design of Neural Nets*, In VLSI Design of Neural Networks, Eds. Ramacher and Ruckert, pp. 2-34, 1991.