MIXED-MODE VLSI IMPLEMENTATION OF FUZZY ART

Marc Cohen, Pamela Abshire and Gert Cauwenberghs

Department of Electrical and Computer Engineering
The Johns Hopkins University, Baltimore, MD 21218-2686
E-mail: {marc,gert}@bach.ece.jhu.edu

ABSTRACT

We present an asynchronous mixed analog-digital VLSI architecture which implements the Fuzzy Adaptive Resonance Theory (Fuzzy-ART) algorithm. Both classification and learning are performed on-chip in real-time. Unique features of our implementation include: an embedded refresh mechanism to overcome memory drift due to charge leakage from volatile capacitive storage; and a recoding mechanism to eliminate and reassign inactive categories. A small scale 1.2μm feature size CMOS prototype with 4 inputs and 8 output categories has been designed and fabricated. The unit cell which performs the fuzzy min and learning operations measures 100 μm by 45 μm. Experimental results are included to illustrate performance of the unit cell.

1. INTRODUCTION

ART classifiers are unique among other classifier models in that they guarantee stable category learning without category proliferation. Fuzzy ART [1] represents inputs and categories in analog form; therefore it is suited to process analog patterns as well as fuzzy representations of discrete-valued patterns under conditions of uncertainty and variability.

While several hardware implementations of related classifier architectures have been developed [2], ART implementations have only recently received attention [5]. Here we describe a modular and scalable VLSI implementation of the Fuzzy ART algorithm that operates on M-dimensional analog input patterns and classifies them as belonging to one of N output categories. M and N can be chosen independently and are constrained only by the die size.

We briefly summarize the Fuzzy ART algorithm and describe some necessary changes that relate to issues of VLSI architecture and technology in the context of our mixed-mode VLSI hardware implementation.

1.1. Summary of The Fuzzy ART Algorithm

For a detailed exposition of the algorithm, we refer to [1] where several variants of the fuzzy ART algorithm have been presented. We did not attempt to implement every alternative; we have chosen a scheme which seems natural and flexible for mixed-mode VLSI circuit implementation. In the following discussion, the M-dimensional inputs will be indexed by the subscript i and outputs (N categories) by the subscript j.

Inputs to the classifier are complement-encoded – that is, each input is encoded both as I_i and $(I_{\max} - I_i)$. This provides automatic "normalization": the sum of all inputs and their complements is always $M \cdot I_{\max}$. For each input and each category, the fuzzy min is computed as the minimum of that input and the stored weight associated with that category:

$$\min(I_i, w_{ij}) + \min(I_{\max} - I_i, v_{ij}) \tag{1}$$

where w_{ij} represents the weight from the i_{th} input to the j_{th} output and v_{ij} the weight from the i_{th} complimentary input to the j_{th} output. Each category output is the sum over inputs (and their complements) of the result of the fuzzy min operation:

$$O_j = \sum_i [\min(I_i, w_{ij}) + \min(I_{\max} - I_i, v_{ij})] \tag{2}$$

All N category outputs are compared with a vigilance parameter ρ, and are normalized by the sum of the weights in that category:

$$\frac{O_j}{\alpha + \sum_i [\mid w_{ij} \mid + \mid v_{ij} \mid]} \tag{3}$$

The output (normalized by the sum of the inputs $M \cdot I_{\max}$) which exceeds ρ and has the maximum normalized value is selected as the winning category. Once a category has been selected, the weights belonging to that category are updated according to the learning rule:

$$\Delta w_{ij} = \beta[\min(I_i, w_{ij}) - w_{ij}] \tag{4}$$

(similarly for v_{ij}). Note that if $I_i > w_{ij}$, $\Delta w_{ij} = \beta(w_{ij} - w_{ij}) = 0$, thus update occurs only if the input is smaller than the stored weight. (Updates can only decrease the stored weight value so it is important that all weights be initialized to their maximum value.) The parameter β controls the learning rate. Carpenter and Grossberg [1] describe a mode of "fast learning" for which $\beta = 1$, and which is used for initial category coding and category recoding. If none of the outputs exceed ρ, a new category is assigned to represent the input using this "fast learning" mode.

1.2. Modifications to the algorithm

As equation (4) indicates, an update only occurs when $\min(I_i, w_{ij}) = I_i$. Furthermore, we anticipate that diode leakage currents onto the weight storage capacitors will cause the weights to decay over time. This decay causes the weights to move in the same direction as the normal update rule. To counteract this problem of category drift we propose the following modification to the learning rule:

$$\Delta w_{ij} = \lambda(I_i - w_{ij}) \tag{5}$$

where

$$\lambda = \begin{cases} 1 & \text{for initial coding and recoding} \\ \beta & \text{when } I_i < w_{ij} \\ \beta_{\text{small}} & \text{when } I_i > w_{ij} \end{cases} \tag{6}$$

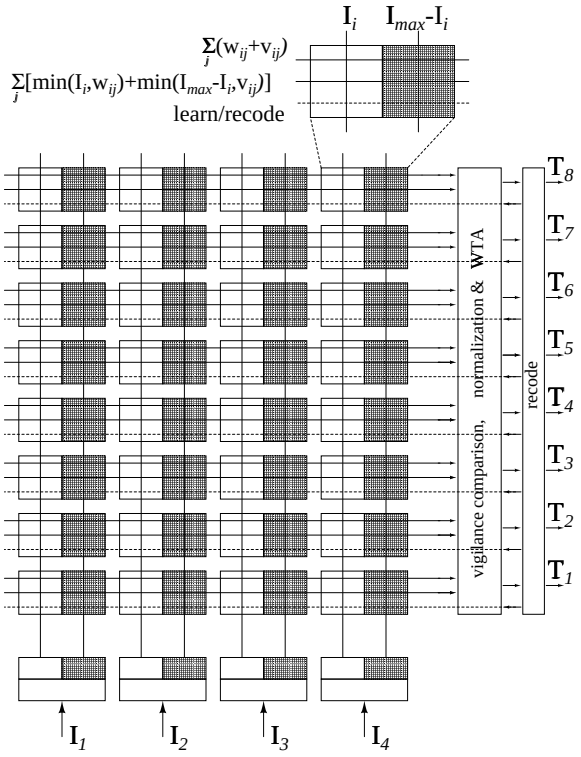

Figure 1: *Chip architecture for $M = 4$ and $N = 8$*

(similarly for vij). This modified rule incorporates the rule of equation (4) and allows a (small) update towards the input in the case where the input exceeds the weight. This ensures refresh of the drifting weights towards the centroid of the input distribution belonging to that particular category. This refresh does not significantly disturb the asymmetry of weight update so long as the learning rate β is sufficiently larger than $\beta_{\rm small}$.

A second modification pertains to the mechanism of assigning new categories. Since N is fixed in the implementation we cannot keep assigning new categories whenever ρ is not exceeded. Instead, it is possible to recode the existing category that has been least frequently chosen in the past. Activity of a category is tracked by a lossy state variable A_j that is updated incrementally whenever that category is selected as the winner. A parallel search for the minimum A_j selects the category to be recoded. Of course, when the category capacity of the chip is not exceeded, existing categories will not be recoded. Recoding can be disabled if desired, for example when it is important to maintain a representation of an infrequently visited category.

2. HARDWARE IMPLEMENTATION

We have adopted a modular, scalable architecture. Figure 1 depicts a 4-input-8-output network. In general, M-dimensional inputs are presented along the bottom of the network and are complement-encoded into $2M$ signals by the input blocks. The dynamic range of the input signals is determined by the choice of $I_{\rm max}$, and inputs may be applied as either currents or voltages. The now complement-encoded inputs are passed to an array of cells each of which computes the fuzzy minimum [see equation (1)] and performs the learning locally for each weight [see equation (5)]. (The shaded cells represent the circuit elements computing with complement encoded signals). The fuzzy min computation is done in the current domain so that it is natural to sum the contributions from all cells in that row. The weights, also represented as currents, are also summed along each row. The summed currents, O_j [see equation (2)] are compared with the vigilance ρ using a simple current comparator and normalized as in equation (3) using a translinear normalizer [4]. If the vigilance criterion is satisfied, the normalized output is passed to a current-mode winner-takes-all circuit [7] which determines the category which best matches the input. A logic "high" signal is broadcast to all the cells along the winning row, and a logic "low" elsewhere, so only the winning row is subject to learning.

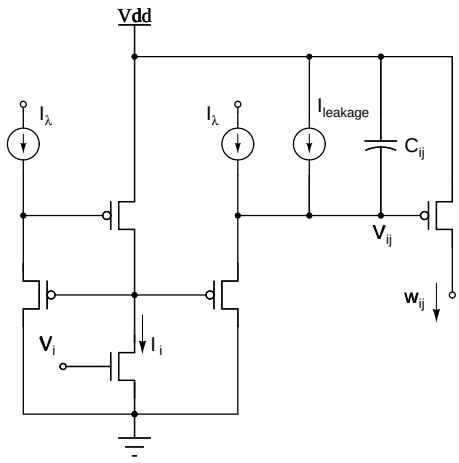

Figure 2: *The learning cell*

Since inputs are explicitly complement-encoded we need only one type of cell in the array, that is one which computes the fuzzy min, performs the weight update and stores the weight. In each cell of the array, weights are stored as voltages V_{ij} on a 1pF capacitor connected to the gate of a p-mos transistor.

Our implementation automatically selects between three different learning rates for each stored weight: a high rate $\beta_{\rm big}$ for initial coding or recoding, a moderate rate β for regular learning, and a low rate $\beta_{\rm small}$ for refresh learning. $\beta_{\rm big}$ is used only when a row is selected for coding/recoding, and it is applied to the entire row which is being coded/recoded so that the resulting weight values are identical to the input. β and $\beta_{\rm small}$, in contrast, are used for incremental updates. When a row is selected as the winner, each cell in that row locally determines whether the input is less than or greater than the stored weight in that cell, and the appropriate learning rate is selected per equation (6).

A schematic of the learning and weight storage cell is shown in Figure 2. It consists of a translinear log domain filter [6] with a 1pF storage capacitor on the gate of the output transistor. The output current w_{ij} is linear in the input current I_i, first-order lowpass filtered with a time constant inversely proportional to I_λ. The circuit is operated as a charge-injection-free charge pump by briefly (1μs) pulsing the learning current I_λ [3]. The filter operates only during the brief learning pulse which is applied to the winning row.

If ρ is not exceeded for any row, then the least frequently used row is selected for recoding: the "high" logic level signal is broadcast to only that row, and the learning rate for that row is temporar-

ily set to $\beta_{\rm big}$.

A small test network comprising 4 inputs and 8 outputs has been fabricated through the MOSIS foundary service in a tiny-chip 1.2μm nwell double-poly technology. Results from a unit cell test-structure are presented here. The unit cell which performs the fuzzy min and learning operations measures 100 μm by 45 μm.

3. FUZZY ART WITH THE MODIFIED LEARNING RULE

3.1. Origins of VLSI Weight Decay

In each cell of the array, the weight current w_{ij} (or v_{ij}) is sourced by a p-mos transistor [see Figure 2] whose gate voltage V_{ij} is set by the storage capacitor. The storage capacitors in our implementation are not completely insulated by gate oxide. Learning is accomplished when current flows to or from the capacitor, and this current is sourced by p-mos transistors. Leakage currents $I_{\rm leakage}$ from these p-type diffusion areas will tend to drive the capacitor voltages toward the positive rail. Thus the voltage on each capacitor C_{ij} will decay linearly from an initial value toward the positive rail ($V_{\rm dd}$). The voltage decay rate can be estimated as follows:

$$\mid \frac{dV_{ij}}{dt} \mid = \frac{I_{\rm leakage}}{C_{ij}} \approx \frac{10^{-14}{\rm A}}{10^{-12}{\rm F}} = 10^{-2}{\rm V/s}. \qquad (7)$$

Since the current is an exponential function of the gate voltage when the transistor is operating in the subthreshold regime, an exponential decay of the currents w_{ij} and v_{ij} will result, with time constant:

$$\tau_{\rm leakage} = U_T C_{ij}/(\kappa I_{\rm leakage}) \approx 4.2{\rm s} \qquad (8)$$

where κ is the subthreshold slope factor and U_T is the thermal voltage. If we choose a presentation rate of $F = 10000$ presentations per second, the weight decay constant γ will be (per presentation):

$$\gamma = 1/(\tau_{\rm leakage} \cdot F) = 2.4 \times 10^{-5} \qquad (9)$$

3.2. Compensation for Weight Decay

During operation of the fuzzy ART algorithm, each category is coded initially via fast learning and then undergoes a period of normal learning during which the category boundaries are enlarged by successive input presentations [1]. Eventually the categories reach a steady state condition wherein their weights describe the distribution of inputs well enough that no further learning is necessary, and ideally all weights should remain constant thereafter. However, weight decay will have the effect of further enlarging the boxes until the category is so large that the vigilance comparison cannot be satisfied for any input. Weight decay acts in the same direction as the normal learning. To avoid this instability it is necessary to counteract the weight decay, and a suitable mechanism is provided by the modified learning rule.

It can be shown that the incremental update has the form of a first-order lowpass filter:

$$\Delta w_{ij} = I_\lambda (I_i - w_{ij}) \qquad (10)$$

[1] As described by Carpenter and Grossberg, the maximum size of the category box is related to the vigilance parameter ρ.

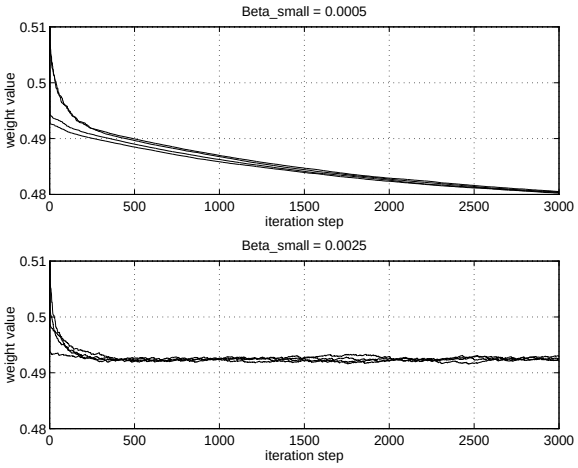

Figure 3: *evolution of weights for different* $\beta_{\rm small}$, $\beta = 0.05$, $\gamma = 2.4 \times 10^{-5}$

This update applies to both normal and refresh learning, but with different values for the learning rate I_λ as in equation (6). Including weight decay, the above equation becomes:

$$\Delta w_{ij} = I_\lambda (I_i - w_{ij}) - \gamma w_{ij} \qquad (11)$$

Weight decay happens continuously for all weights, whereas refresh learning occurs only during the learning pulse and only for the weights belonging to the winning category. On average, for N active categories, each of which is selected with equal probability, refresh learning occurs only once for every N input presentations. To maintain stable categories, the refresh learning must balance the weight decay:

$$\Delta t \cdot F \cdot I_{\beta\,\rm small} \cdot E[I_i - w_{ij}]/N \geq \gamma E[w_{ij}] \qquad (12)$$

where Δt is the length of the learning pulse, F is the frequency of presentation and E denotes the expectation operator. This constraint is satisfied if $\gamma \cdot N \ll I_{\beta\,\rm small} \cdot \Delta t \cdot F$. In addition, the asymmetry of the original weight update (no decay and no refresh) is preserved if $I_\beta \gg I_{\beta\,\rm small}$. The constraints on the learning rates can be summarized as follows:

$$I_\beta \gg I_{\beta\,\rm small} \gg \gamma \cdot N/(\Delta t \cdot F) \qquad (13)$$

This is easily satisfied for anticipated values of leakage currents, presentation frequency and learning currents.

Simulations have been performed [see Figure 3] to verify that the refresh learning compensates for weight decay and doesn't affect the asymmetry or stability of the original algorithm. When $I_{\beta\,\rm small}$ is set correctly, the weights fluctuate about their appropriate values. When $I_{\beta\,\rm small}$ is too small, the weights decay until the category is too large for any input to satisfy the vigilance criterion, and a new category is erroneously created. For the results in Figure 3, two inputs were each uniformly distributed in [0.49, 0.51] and the vigilance during training was set at 0.8, so that one category described the input distribution. The top plot shows the decay of the weights when $I_{\beta\,\rm small}$ is too small. At the given vigilance, a new category will be created when these weights decay to 0.4. The bottom plot shows that the weights fluctuate about the correct value for an appropriate value of $I_{\beta\,\rm small}$.

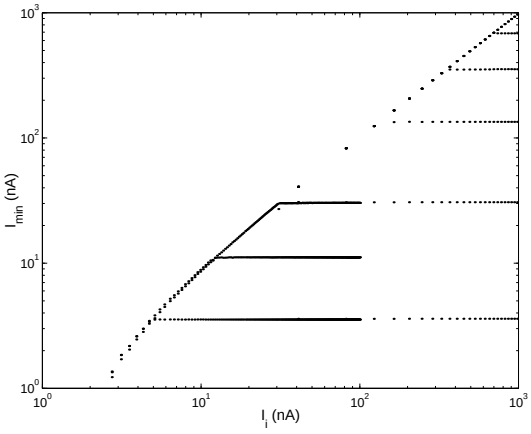

Figure 4: *Measured Fuzzy-min Computation*

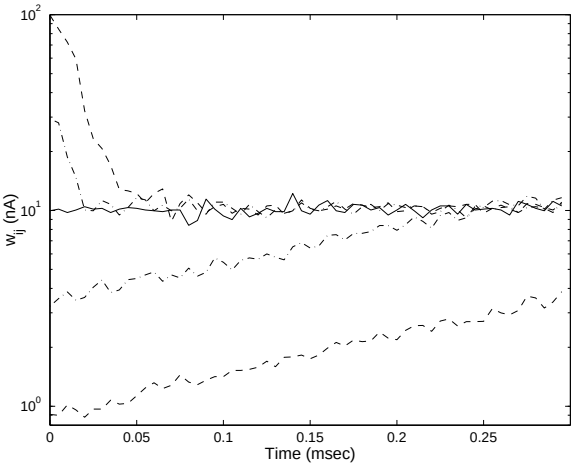

Figure 5: *Measured Normal and Refresh Learning*

4. TEST RESULTS: FUZZY MIN AND LEARNING

We performed experimental tests on a fabricated prototype to validate performance. The results presented here were obtained from an isolated cell so as to characterize the fuzzy min computation and learning

With learning disabled, we manually set the weight current w_{ij} then ramped the input current I_i from a value less than w_{ij} to a value greater than w_{ij}. When the input is less than the stored weight, the output follows the input. Once the input exceeds the stored weight, the output remains constant at the stored weight value. Figure 4 plots the measured results for I_i and w_{ij} ranging from about $3nA$ to $1000nA$ clearly demonstrating $\min(I_i, w_{ij})$.

To demonstrate normal learning (I_β), w_{ij} was initially held fixed at a value greater than I_i and then released to adapt down to I_i. This normal learning proceeded at a fast rate (time constant approximately $35\mu sec.$). For refresh learning ($I_{\beta\text{small}}$), w_{ij} was initially held fixed at a value less than I_i and then released to adapt up slowly to I_i (time constant approximately $200\mu sec.$) Figure 5 shows the measured results for an input current of $10nA$ and a range of initial weight values. The decaying traces indicate w_{ij} adapting from $100nA$ ($30nA$) down to $10nA$ with normal learning and the slowly increasing traces indicate w_{ij} adapting from $1nA$ ($3nA$) up to $10nA$ with refresh learning.

5. CONCLUSIONS

We have implemented an asynchronous mixed-mode CMOS VLSI system capable of classifying and learning in real-time. We included in our implementation an addition to the learning rule that gives us the capability to refresh the analog weights and a mechanism for recoding categories which are infrequently used. Both of these new features can be disabled if required. We have verified by simulation that our modified learning rule preserves both the asymmetry and the stability of the original rule in the presence of weight decay in an analog implementation. We have also experimentally verified both the fuzzy-min computation and learning capability of the fabricated unit cell.

Acknowledgements

This work was supported by the Multi-University Research Initiative (ARPA and ONR Grant # N00014-95-1-0409). Chip fabrication was provided by MOSIS. The second author was supported by an NSF Graduate Fellowship.

6. REFERENCES

[1] Gail A. Carpenter, Stephen Grossberg and David B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks,* vol. **4**, pp 759-771, 1991.

[2] Gert Cauwenberghs and Volnei Pedroni, "A Charge-Based CMOS Parallel Analog Vector Quantizer", *Advances in Neural Information Processing Systems,* vol. **7**, Ed: D. S. Touretzky, Morgan Kauffmann, San Mateo, CA, 1995.

[3] Gert Cauwenberghs and Ammon Yariv, "Fault-Tolerant Dynamic Multilevel Storage in Analog VLSI," *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing,* vol. **41** No. 12, December 1994.

[4] Barry Gilbert, "Current-mode Circuits From a Translinear Viewpoint A tutorial," in *In Analogue IC Design: the current-mode approach.* Eds: C. Toumazou, F. J. Lidgey and D. G. Haigh. IEE Circuits and Systems Series 2, pp 11-91, 1990.

[5] T. Serrano-Gotarredona, B. Linares-Barranco and J. L. Huertas, "A Real Time Clustering CMOS Neural Engine", *Advances in Neural Information Processing Systems,* vol. **7**, Ed: D. S. Touretzky, Morgan Kauffmann, San Mateo, CA, 1995.

[6] W. Himmelbauer, PM Furth, PO Pouliquen, and AG Andreou, "Log-domain Filters in Subthreshold MOS", JHU/ECE Technical Report 96-03.

[7] J. Lazzaro et. al., "Winner-Take-All networks of O(n) complexity," in *Advances in Neural Information Processing Systems,* vol. **1**, Ed: D. S. Touretzky, pp 703-711. Morgan Kauffmann, San Mateo, CA, 1988.