ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

VHDL-AMS MODELING OF SELF-ORGANIZING NEURAL SYSTEMS

Jose-Alejandro Lopez-Alcantud[†] *and Tom Kazmierski*[‡]

[†]Universidad Politecnica de Cartagena
Paseo Alfonso XIII, 48, E-30203 Cartagena, Spain

[‡]University of Southampton
Southampton SO17 1BJ, England

ABSTRACT

This contribution presents VHDL-AMS models and simulation results for a complex, self-organizing neural system based on the adaptive resonance theory. Such neural systems exhibit both discrete and continuous dynamic behavior and consist of a large number of analog equations, a digital controller with analog and digital feedback paths resulting in the complexity that would prohibit analysis with conventional mixed-signal simulation tools.

1. INTRODUCTION

The advances in the integrated circuit technology have led to the growing popularity and a decrease in cost of mixed-signal ASICs (Application Specific Integrated Circuits) that comprise both analogue and digital circuit blocks. VLSI implementations of mixed-signal systems with very large analogue parts are extremely difficult due to the complexity and prohibitive times of circuit-level simulation. The development of CAD tools in recent years has concentrated mainly on automatic design of digital circuits while the automation of analogue designs remains largely a heuristic and a labor-intensive process. Complexity is the primary reason for slow progress in the development and application of artificial neural networks in the VLSI technology. The emergence of the IEEE standard for VHDL 1076.1, informally known as VHDL-AMS [1] has given a new impetus to the advancement of analogue and mixed-signal synthesis methodology [2]. VHDL-AMS extends the modeling capabilities of VHDL to enable descriptions of high-level systems with continuous dynamic behavior that can be specified by means of algebraic or ordinary differential equations. The first commercial VHDL-AMS simulators [3, 4] are already available. It is envisaged that the evolution of automated analogue synthesis based on VHDL-AMS descriptions will lead to the development of CAD tools capable to synthesize complex analogue-digital systems automatically without the need to resort to circuit-level verification. In this paper we present a technique for developing VHDL-AMS models of complex self-organizing neural systems capable to categorize analogue input patterns for further recognition. Self-organizing neural networks comprise substantial analogue processing parts combined with digital, event-driven control (Fig 1.). The system we have chosen as a case study is an Adaptive Resonance Theory-based neural network, proposed by Carpenter and Grossberg [5]). Neural networks based on this theory have been used in a wide range of applications, such as fuzzy control systems [9] or image recognition [10, 11]. ART2 has been used for chemical plants monitoring [12] and in biomedical applications [13, 14]. We briefly describe the system's mathematical model, its VHDL-AMS implementation and simulation results obtained with Mentor VHDL-AMS Design Station [3].

2. VHDL-AMS MODEL

In order to model the ART2 neural network model by means of a behavioral description using VHDL-AMS, we have divided it into several modules as it is shown in Fig. 1. There are four modules that fit to the processing layers of ART2 [5, 6]: F0, F1 and F2 layers and the orientative system. But from an electronic implementation point of view, we have added two modules to this architecture: a memory module (where the weights are stored and updated) and a control unit (to synchronize all the modules in the input stage, resonating stage and learning stage). It is important to emphasize that the description of this system is mixed because some analogue modules (layers F0, F1 and F2 as well as the orientative system) are included together with a digital control unit.

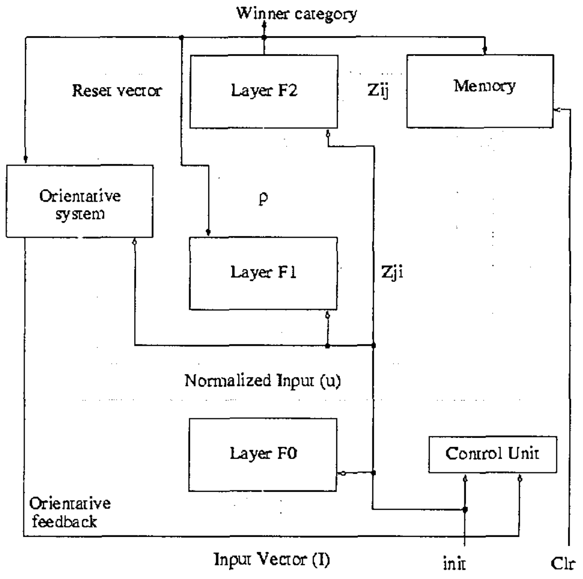

Figure 1. Block diagram of the ART2 VHDL-AMS model

The self-learning algorithm implemented in our model is shown in Fig. 2. In the following explanation we will consider M inputs and N categories in the ART2 neural network.

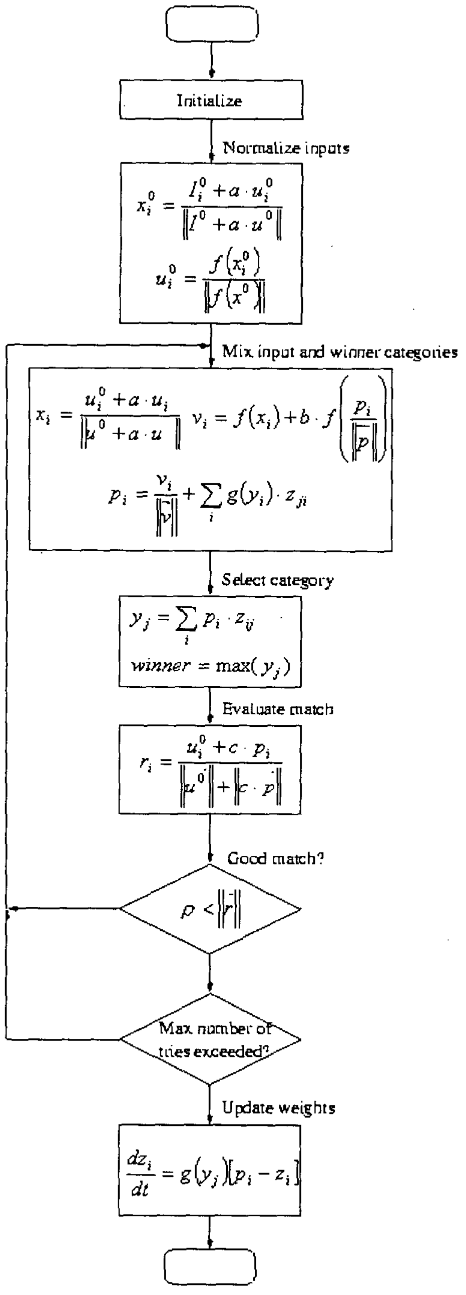
Initialize
Normalize inputs
$x_i^0 = \frac{I_i^0 + a \cdot u_i^0}{\|I^0 + a \cdot u^0\|}$
$u_i^0 = \frac{f(x_i^0)}{\|f(x^0)\|}$
Mix input and winner categories
$x_i = \frac{u_i^0 + a \cdot u_i}{\|u^0 + a \cdot u\|}$
$v_i = f(x_i) + b \cdot f\left(\frac{p_i}{\|p\|}\right)$
$p_i = \frac{v_i}{\|v\|} + \sum_i g(y_i) \cdot z_{ji}$
Select category
$y_j = \sum_i p_i \cdot z_{ij}$
$winner = \max(y_j)$
Evaluate match
$r_i = \frac{u_i^0 + c \cdot p_i}{\|u^0\| + \|c \cdot p\|}$
Good match?
$\rho < \|r\|$
Max number of tries exceeded?
Update weights
$\frac{dz_i}{dt} = g(y_j)[p_i - z_i]$

Figure 2. Data flow of the ART2 VHDL-AMS model

2.1 Input layer F0

The input layer performs the pre-processing and normalization of the input vector **I** = (I_1, …, I_i, …, I_M) using the equations shown in Fig 2. A linear pruning function $f(x)$: $\{x,\ x>\theta;\ 0,\ x\leq \theta\}$ is used to eliminate noise from input information, and the L2 norm is used for normalization. The noise threshold θ and the parameter a are customizable depending on the application.

2.2 Inner layer F1

The inner layer processes the normalized pattern **u** = (u_1, …, u_i, …, u_M). generated by F0 and mixes it with the learned pattern produced by F2 as it will be explained later. The mixing ratio is determined by two parameters: a and b. The adaptive resonance operation in F1 is controlled by the signals generated by the control unit (Fig. 1) as illustrated in the following VHDL-AMS description:

```
ARCHITECTURE behavior OF f1 IS
  ...
BEGIN
 -- If there is a new input or the orientative system reset the winner
 -- category in F2, the nodes values are set to 0 (including
 -- intermediate nodes w and q)
 IF (clear='1') OR (reset='1') USE
   x1==0.0; u1==0.0; w1==0.0; v1==0.0; p1==0.0; q1==0.0;
  ...
 -- When the control unit enables F1, update nodes using ART2 eqns.
 ELSIF (enable='1') USE
   w1==u01+a*u1;
   x1==w1/norm(w1,w2,w3);
   v1==f(x1,threshold)+b*f(q1,threshold);
   u1==v1/norm(v1,v2,v3);
   q1==p1/norm(p1,p2,p3);
  ...
 -- evaluate p1, the output from of this layer according to
 -- top-down weights and winner category
   IF (win=1) USE
     p1==u1+d*ztd1_1;
   ...
   ELSIF (win=2) USE
     p1==u1+d*ztd2_1;
   ...
   ELSE
     p1==u1;
   ...
   END USE;
  ...
 ELSE
   x1'dot==0.0; u1'dot==0.0; w1'dot==0.0; v1'dot==0.0;
   p1'dot==0.0;q1'dot==0.0;
  ...
 END USE;
END ARCHITECTURE behavior;
```

The winner category win is a signal evaluated by the output layer and the top-down weights ztd*j*_*i* (where i=1,…, M and j=1,…, N) are provided by the memory.

2.3 Output layer F2

The output layer implements a winner-takes-all algorithm to select a category J for the input pattern. This is done by iterative comparison of learned patterns stored in the memory of adaptive weights z_{ij} with the filtered pattern $\mathbf{p}= (p_1, \ldots, p_i, \ldots, p_M)$ obtained from F1. The following sample of VHDL-AMS illustrates the behavior of F2:

```
ARCHITECTURE behavior OF f2 IS
  ...
BEGIN
 -- If there is a new input or the orientative system reset the winner
 -- category, set the node values (activities) to 0
 IF (clear='1') OR (reset='1') USE
  y1==0.0;
  ...
 ELSE
 -- Evaluate node activities using p from F1 and bottom-up weights.
  IF (reset1='1') USE
   y1==p1*zbu1_1+p2*zbu2_1+p3*zbu3_1;
  ELSE
   y1==0.0;
  END USE;
  ...
  END USE;
 END USE;
 -- The winner ("win") is the node with the highest activity
 win<= 0 WHEN clear='1' OR reset='1' ELSE
        1 WHEN enable='1' AND reset1='1' AND
          y1>=y2 AND y1>=y3 AND y1>=y4 AND y1>=y5 ELSE
  ...
END ARCHITECTURE behavior;
```

2.4 Orientative system

The comparison between the learned pattern and the input pattern results in the calculation of a matching vector $\mathbf{r}= (r_1, \ldots, r_i, \ldots, r_M)$. A vigilance parameter ρ is used to monitor the matching level. If $\rho > \| \mathbf{r} \|$,. i.e. no good match is found, the control unit activates a reset signal to disable the currently selected F2 node and initiate a new search cycle in layer F2.

In VHDL-AMS terms this is achieved using concurrent signal assignments and simultaneous statements in the architecture of the orientative system:

```
ARCHITECTURE behavior OF orient IS
  ...
BEGIN
 r1==(u01+c*p1)/(norm(u01,u02,u03)+norm(c*p1,c*p2,c*p3));
  ...
 norma_R==norm(r1,r2,r3);
 PROCESS (clear, enable) -- set category resets
 BEGIN
  IF (clear='1') AND (clear'LAST_VALUE='0') THEN
    reset1<='1';
  ...
  ELSIF (enable='1') AND (enable'LAST_VALUE='0') THEN
   IF reset1='1' THEN
     reset1<=ori(rho,norma_R,win,1);
   END IF;
  ...
  END IF;
 END PROCESS;
 -- set reset if no category matches...
 reset<= reset1='0' AND win=1 OR reset2='0' AND win=2 ...
  ...
END ARCHITECTURE behavior;
```

2.5 Memory

The memory stores the adaptive weights (z_{ij} and z_{ji}) for all the nodes in F2 and updates the weights when a winner node is found in F2, i.e. when the selected category and the input vector manifest a good match. We have tested our model with a fast learning [7] and a slow learning [6] mode of operation. The slow learning mode is regarded to be more reliable and accurate and also it simulates well human or animal learning behavior. However, VHDL-AMS models of the slow learning mode are vastly more complex than the fast learning one since the former is based on dynamic solution of a large set of ordinary differential equations. The slow learning mode is represented in VHDL-AMS as follows:

```
ARCHITECTURE behavior OF memory IS
  ...
BEGIN
 IF (clear='1') USE
  zbu1_1==1.0/((1.0-d)*sqrt(M));
  ...
  ztd1_1==0.0;
  ...
 ELSIF (slow='1') USE
  IF win=1 USE
   zbu1_1'dot==d*(1.0-d)*(u1/(1.0-d)-zbu1_1);
   zbu1_2'dot==0.0;
   ...
   ztd1_1'dot==d*(1.0-d)*(u1/(1.0-d)-ztd1_1);
   ...
   ztd2_1'dot==0.0;
   ...
  ELSE
   zbu1_1'dot==0.0;
   ztd1_1'dot==0.0;
   ...
  END USE;
 ELSE
  zbu1_1'dot==0.0;
  ztd1_1'dot==0.0;
  ...
 END USE;
END ARCHITECTURE behaviour;
```

3. EXPERIMENTS

We have tested our model using several different system sizes. The results shown in Fig.3 represent the learning behavior of a system with three input signals and five possible output categories. The system comprises in total 64 linear and 6 non-linear algebraic equations, and 30 non-linear ordinary differential equations that provide the learning model for the 30 weights.

Table 1 shows a comparison between the categorization results obtained from our VHDL-AMS simulations with the categorization calculated by the dedicated ART2 simulator available from Boston

University [8] using the same input waveforms and parameter values. With the exception of pattern 8 in Table 1, both results are identical.

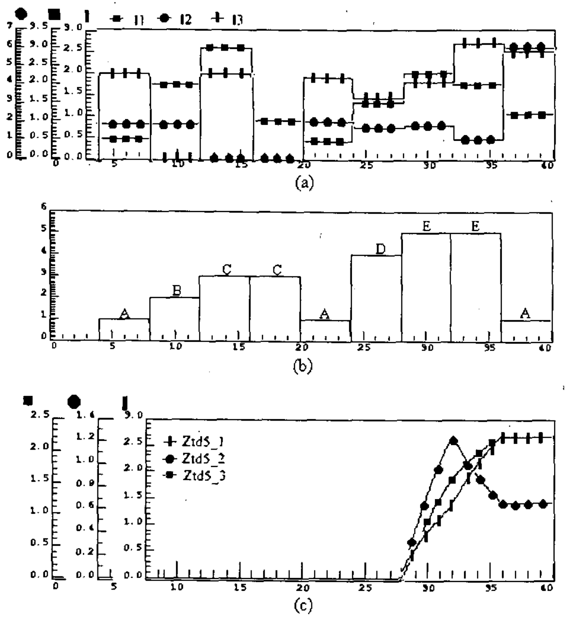

Figure 3. Simulation results. a) simulated input patterns b) winner categories in F2. c) top-down weights.

Input Pattern	Values			Simul. time (s)	Category	
	I_1	I_2	I_3		*Dedicated simul. [8]*	*VHDL-AMS*
1	0.50	2.00	2.00	207.53	A	A
2	2.00	2.00	0.00	221.54	B	B
3	3.00	0.00	2.00	468.26	C	C
4	1.00	0.00	0.00	101.15	C	C
5	0.50	2.00	1.90	211.40	A	A
6	1.50	1.65	1.45	330.96	D	D
7	2.30	1.80	1.80	302.19	E	E
8	2.00	1.00	2.70	260.45	C	E
9	1.20	6.00	2.50	265.49	A	A

Table 1. Input pattern categorization.

4. CONCLUSION

We have demonstrated the capability of VHDL-AMS to model accurately very complex systems with mixed, discrete and continuous behavior. The complexity of neural systems based on the adaptive resonance theory is the effect of both the system size and feedback paths in the discrete and continuous domains. It is envisaged that this work will aid automated synthesis of self-organizing neural systems in the VLSI technology. Prohibitive simulation times for such systems with conventional mixed-signal simulation techniques make automated synthesis very difficult. Automated analog and mixed-signal synthesis systems based on VHDL-AMS, analogous to their existing digital VHDL counterparts, will soon emerge. It will therefore be essential to provide software aids able to verify complicated mixed-signal systems on the behavioral level before mapping them into silicon directly from their VHDL-AMS descriptions, without the need for exhaustive circuit-level simulations.

5. REFERENCES

[1] Design Automation Standards Committee of the IEEE Computer Society, *IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes).* Draft IEEE Std 1076.1. August 1, 1998.

[2] Doboli A, Vemuri R., *A VHDL-AMS compiler and architecture generator for behavioral synthesis of analog systems*, Proc. of DATE'99, Munich 9-12 March'99, pp. 338-345.

[3] Mentor Graphics. *VHDL-AMS Design Station User Manual.* Software revision v1.0_1.1. March 1999.

[4] Analogy. *VeriasHDL simulator.* 1999.

[5] Carpenter G.A. and Grossberg S. *A massively parallel architecture for a self-organizing neural pattern recognition machine.* Computer vision graphics image processing, Vol. 37, n 54, 1987.

[6] Carpenter G.A. and Grossberg S. *ART2: Self-organizing of stable category recognition codes for analog input patters.* Applied Opties, Vol. 16, n 23, Diciembre, 1987.

[7] Carpenter, Gail A., Grossberg, Stephen and Rosen, D.B. *ART2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition,* Neural Networks, 1991, Vol. 4, pp.493-504.

[8] Gaudiano, Paolo. *ART2 simulator.* May 14, 1990.

[9] Lin Ch.-J, Lin Ch.-T *An ART-based fuzzy adaptive learning control network,* 1997.

[10] Hartigan J.A. *Clusters Algorithms,* John Wiley k Sons, IRF Scientific Report 214, 1993, 'The Freja scientific satellite", IRF Kiruna third edition, 1975.

[11] Healy M.J, Caudell T.P. *Acquiring rule sets as a product of learning in a logical neural architecture.* IEEE Transactions on neural networks, Vol. 8, n. 3, pp. 461-473, 1997.

[12] Whiteley J.R., Davis J.F., Mehrotra A, Ahalt S.C. *Observations and problems applying ART2 for dynamic sensor pattern interpretation.* IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and humans, Vol. 26, n 4, pp. 423-237, 1996.

[13] Suzuki Y, Ono K. *Personal computer system for ECG ST-segment recognition based on neural networks.* Medical & Biological Engineering & Computing, Vol. 30, pp. 2-8, 1992.

[14] Suzuki Y. *Self-organizing QRS-wave recognition in ECG using neural networks.* IEEE Transactions on Neural Networks, Vol. 6, n 6, pp. 1469-1477, 1995.