Hybrid optoelectronic adaptive resonance theory neural processor, ART1

Thomas P. Caudell

For industrial use, adaptive resonance theory (ART) neural networks have the potential of becoming an important component in a variety of commercial and military systems. Efficient software emulations of these networks are adequate in many of today's low-end applications such as information retrieval or group technology. But for larger applications, special-purpose hardware is required to achieve the expected performance requirements. Direct electronic implementation of this network model has proven difficult to scale to large-input dimensionality owing to the high degree of interconnectivity between layers. Here, a new hardware implementation design of ART1 is proposed that handles input dimensions of practical size. It efficiently combines the advantages of optical and electronic devices to produce a stand-alone ART1 processor. Parallel computations are relegated to free-space optics, while serial operations are performed in VLSI electronics. One possible physical realization of this architecture is proposed. No hardware has as yet been built.

Key words: Free-space optical processing, neural networks, adaptive resonance theory, ferroelectric liquid-crystal spatial light modulators, VLSI analog processing.

1. Introduction

Adaptive resonance theory (ART) networks, a class of neural networks developed by Carpenter and Grossberg[1] in 1987, are used to learn stably the classification codes for streams of input patterns. Clustering is the basic function performed by an ART network.[2] These networks generate clusters in a self-organizing manner, detect and treat novel patterns, and learn generalizations of the patterns in a relatively small number of presentation cycles.[3,4] In Ref. 4 the theoretical upper limits for the number of presentation epochs necessary to stabilize the learning of ART are all large compared with those observed in practice. For example, an operational system with over 1000 distinct complex input patterns always converges in fewer than ten epoches. We focus on the specific form of ART that handles binary-input patterns known as ART1.

Currently ART1 networks are beginning to be used in a growing number of industrial applications. At The Boeing Company, Caudell *et al.*[5] used ART1 networks in an information retrieval system for the search and recall of similar engineering designs, which is called the group-technology problem. Baloch and Waxman[6] used an ART1 network in a modular way to control the behavior of mobile robots.

In all these applications, ART networks are implemented in software simulations on conventional computer systems, either in serial or in parallel. In terms of execution time, the performance of these simulations is acceptable either when small input patterns are used or when the applications have no restrictive turnaround-time requirements. The temporal performance is usually not an issue when classification or generalization performance is the focus of research. However, when the simulations grow to meet the requirements of problems found in the real world, the brick wall of simulator performance is quickly encountered. Thus we need to consider physical implementations of these network models.

Such an idea is not new. In the recent past, pure electronic implementations of ART1 components and systems have been proposed. Kaburlasos *et al.*[7] have constructed a microcontroller-based ART1 coprocessor for IBM PC's. Two parallel microcontrollers simulate the solutions to the nonlinear differential equations for two of the layers in the network. Software was necessary to complete the functioning ART1 network. The system is limited to 64 input and 40 output neurodes. Tsay and Newcomb[8] pre-

The author is with Research and Technology, Boeing Computer Services, The Boeing Company, P.O. Box 24346, Seattle, Washington 98124-0346.

Received 20 August 1991.
0003-6935/92/296220-10$05.00/0.

sented a VLSI implementation of the two major subsystems in the ART1 network: the short-term and the long-term memories. The nonlinear differential equations are implemented by using complementary metal-oxide semiconductor (CMOS) devices and circuits. A small system with four inputs and four outputs was layed out and simulated by using Magic, a VLSI simulation and analysis software tool. The high interconnection density needed for the ART1 model and the diversity of synapses within the architecture limit the size of the input vector in purely electronic implementations that use parallism.

Optics and electronics have also been previously combined to address this interconnection problem. Many of the limitations found in pure electronic implementations of neural networks were thus removed. Psaltis *et al.*[9] implemented an optoelectronic version of the Hopfield network by using an optical matrix–vector multiplier. Caulfield and Armitage[10] implemented a pseudo-optical version of the ART1 network that used the optical associative memory model; pseudo in that it does not learn in real time. Recently Wunsch *et al.*[11] proposed and tested the first hybrid optoelectronic ART1 system that used a VanderLugt, or 4*F*, optical correlator.[12] In Ref. 11 an algorithmic formulation of the ART1 network was implemented. The optics performed the bulk of the computations needed for the algorithm by using the intensity of the zero-shift correlation peak to represent vector dot products. In this system, all other calculations needed for the network algorithm are performed in a separate digital computer.

The following study takes the concept of an algorithmic implementation several steps further. After the details of the ART1 neural network algorithm are reviewed in Section 2, the design of an optoelectronic architecture that implements the algorithm completely, without requiring a separate digital computer, is presented in Section 3. The design of one possible physical realization of this optoelectronic architecture is then presented in Section 4, and a method for using this device as a part of a larger macrocircuit of ART1 network modules is discussed in Section 5.

2. Algorithmic Form of ART1

The ART1 neural network model is canonically represented by a coupled set of ordinary nonlinear differential equations.[1] If appropriate restrictions are made on the relationship between the dynamical time constant, the learning rate, and the length of time the input pattern is stable on the network input neurodes, then this system of equations reduces to a procedural algorithm.[2] The dynamical time constants are required to be much smaller than the learning rate, which in turn are much smaller than the stable presentation time. These restrictions lead to a fast learning mode of operation in which the learned weights in the system reach asymptotic stability before a new input pattern is presented. The impact of these restrictions on the implementation of ART1 is enormous: the computational steps of the algorithm can now be mapped directly onto an algorithmic processor. Under these special conditions for this model, we need not become embroiled in the analog implementation of a complex dynamical system. The mapping of the ART1 algorithm onto a specific processor depends on many details, not the least of which is the individual computational steps required in the algorithm. To clarify these statements, the remainder of this section describes the ART1 algorithm. Section 3 discusses the partitioning of this algorithm between the optical and the electronic components.

Figure 1 is a flow chart for the ART1 algorithm and includes the main pattern presentation loop. Table 1 lists the functional operations. The network is trained through repetitive presentation of a set of training patterns. This algorithm autonomously

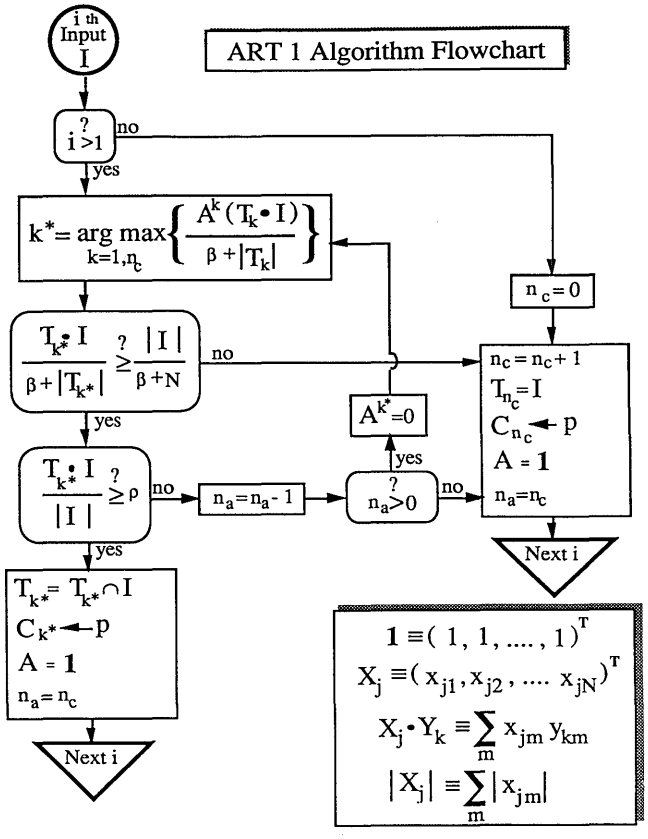

Fig. 1. Flowchart that embodies the processing of an ART1 neural network: i is the index of the current input vector **I** being presented to the network; arg max is a function that returns the index of the maximum element in its list argument, effectively implementing the competition within the F2 layer; $\mathbf{T}_k$ is a typical template in the set of all n_c templates; A^k is a list of binary flags that determine which templates participate in the competition; β is a small constant that biases the search process toward smaller templates; k^* is the current index of the winning F2 neurode; N is the total number of F1 neurodes and is the dimensionality of the input vector, ρ is the vigilance factor, ranging between 0 and 1; n_a is the number of F2 neurodes still active in the competition; p is a distinguishing name or label for **I**; C_{k^*} is the membership roster for cluster k^* containing the names of all of the input vectors that were attracted to $\mathbf{T}_{k^*}$; **1** is the unit vector.

Table 1. List of Principal Computational Operations Required to Performing the ART1 Algorithm[a]

No.	Operation	Electronics	Optics	%
1[b]	$\mathbf{I}$		√	
2[b]	$\mathbf{T}_k$		√	
3[c]	$\mathbf{T}_k \cdot \mathbf{I}$		√	91
4[c]	$\lvert\mathbf{T}_k\rvert$		√	8
5[c]	$\mathbf{T}_{k^*} = \mathbf{T}_{k^*} \cap \mathbf{I}$		√	<1
6[c]	$\lvert\mathbf{I}\rvert$		√	<0.1
7[c]	$\mathbf{T}_{n_c} = \mathbf{I}$		√	<0.1
8	arg max{ }	√		<0.01
9	$\{A^i = 1; i = 1, n_c\}$	√		<0.01
10	$C_{k^*} \leftarrow p$	√		<0.01
11	$\geq$	√		<0.01
12	$\times$	√		<0.01
13	/	√		<0.01
14	+	√		<0.01

[a]These operations can be partitioned naturally between electronic and optical implementations, as indicated by the check marks. The last column gives an approximate percentage of execution time in the ART1 algorithm.

[b]Representation of information as a light intensity or transmissivity.

[c]Assuming that $N_{\text{bit}} = 1024$ and $n_c = 100$.

places binary input patterns (vectors) into clusters. Each cluster is represented in neural memory as a template prototype. A template is an abstraction of the patterns in a cluster, and it is formed and modified during the learning (training) process. The number of clusters discovered by the network is determined by the underlying structure or semantics of the training set of input patterns, by the order of presentation of input patterns, and by a system-level threshold called the vigilance factor. During testing of the trained network, neural memories or templates are directly recalled when examples from known clusters are presented. On the other hand, the learning process remains forever plastic in this model, even after training, which permits new or novel input patterns to be learned without disturbing old memories.

The algorithm proceeds as shown in Fig. 1. A binary input pattern $\mathbf{I}$ is presented to the system. If this is the first presentation of the first pattern, then this pattern becomes the first template, $\mathbf{T}_1$. That is, $\mathbf{I}$ gets copied into $\mathbf{T}_1$, and the number of clusters n_c is set to 1; $\mathbf{T}_{k^*} \leftarrow \mathbf{I}$, where $k^* = 1$ (Table 1, operation 7). After this step, the system is ready for the next input pattern.

Otherwise, a search must determine if $\mathbf{I}$ is close enough to any one of the existing templates. A best-first search is performed in the order of descending value of a normalized dot product $(\mathbf{I} \cdot \mathbf{T}_{k^*})/(\beta + |\mathbf{T}_{k^*}|)$, where k^* is the index of the template under current consideration in the search (Table 1, operations 3, 4, 13, and 14). (The β factor is a small constant to help bias the choice toward smaller templates; it is numerically equal to the reciprocal of N, the number of binary components of the input vector.) The index k^* is obtained by

$$\arg\max_{k=1,n_c} \left[\frac{A^k(\mathbf{I} \cdot \mathbf{T}_k)}{\beta + |\mathbf{T}_k|}\right],$$

where the set of flags A^k is used to select the templates that are permitted to compete for a match with the input pattern; $A^k = 1$ for active templates, while $A^k = 0$ for inactive templates (Table 1, operations 8 and 12). The arg max function returns the index of the maximum element in the argument list. Within the presentation time of a pattern, all templates start off active, but become labeled inactive if, after being selected in the best-first search, they fail the close-enough test to $\mathbf{I}$. This test requires that $\mathbf{I}$ fall within a region of feature space around the template $\mathbf{T}_{k^*}$ to be considered a member of cluster k^*. This region is define by the inequality $(\mathbf{I} \cdot \mathbf{T}_{k^*})/|\mathbf{I}| \geq \rho$, where $\rho \in [0, 1]$ is the vigilance factor for the network (Table 1, operations 6, 11, and 13). A ρ value of approximately 1 gives a network with high discrimination, which forms many clusters. A value of approximately 0 gives a network with low discrimination, which forms few clusters. If the vigilance test is not satisfied for the cluster k^*, then a reset occurs; $A^{k^*} = 0$, and the search continues.

If the vigilance test is satisfied, then the network is in a state of resonance and the winning template $\mathbf{T}_{k^*}$ is updated through a simple learning process. The new template gets the logical bit-wise AND of the old template and the input pattern; $\mathbf{T}_{k^*} \leftarrow \mathbf{T}_{k^*} \cap \mathbf{I}$ (Table 1, operation 5). This learning model in ART1 is a form of Hebbian learning,[13] which is found in many of today's self-organizing neural models. In addition to this operation, the input pattern name is added to the membership set for the k^* cluster, and the active cluster flags $\{A^k\}$ are restored to unity (Table 1, operations 9 and 10). That is, $C_{k^*} \leftarrow p$ and $\{A^k = 1, 1 \leq k \leq n_c\}$, where p is the name of the current input pattern and C_{k^*} is the set of names of input patterns, or the roster, of the members of the k^*th cluster. The name can be a number or an ASCII sequence. Constructing the rosters for clusters provides information useful in many practical applications. After this step, the system is ready for the next input pattern.

If none of the existing clusters pass the vigilance test for input pattern $\mathbf{I}$, then a new cluster is formed. That is, the novel pattern $\mathbf{I}$ gets copied into $\mathbf{T}_{k^{\text{new}}}$, and the number of clusters n_c is incremented by one; $\mathbf{T}_{k^{\text{new}}} \leftarrow \mathbf{I}$, where $k^{\text{new}} = nc + 1$ (Table 1, operation 7). The novel input pattern name is added to a new roster for the k^{new}th cluster, and the active cluster flags $\{A^k\}$ are restored to unity (Table 1, operations 9 and 10). That is, $C_{k^{\text{new}}} \leftarrow p$ and $\{A^k = 1, 1 \leq k \leq n_c\}$. After this step, the system is again ready for the next input pattern.

As is evident in the above discussion, the ART1 algorithm has a mixture of both serial and parallel operations. The vigilance test and the search process remain serial-operations independent of any particular hardware implementation scheme. The

best that hardware can do for ART1 is to drive the net execution time in the parallel portions of the algorithm to as near zero as possible. Section 3 discusses the partitioning of these operations between electronic and optical hardware to achieve high performance.

3. Optoelectronic Architecture

The second column of Table 1 summarizes the mathematical operations found in the ART1 algorithm. As we can see, several of these operations are matrix–vector multiplications. In fact, the major computational load arises from the computation of the vector dot products and norms. Fortunately, optical computing systems can perform these types of operations efficiently, as is indicated in the fourth column of the table. It is shown in this section that all vector and matrix–vector operations can be performed in parallel by using a modification of a well-known optoelectronic processor. To date, no hardware prototypes of this system have been constructed to our knowledge, although research is progressing in that direction.

A. Optical Matrix–Vector Multiplier

In 1985 Farhat and Psaltis[14] proposed and demonstrated the use of the optical matrix–vector multiplication processor (OMVMP) in a neural network system. Figure 2 shows the basic components of this conceptually simple system. The processor performs a multiplication between a vector whose components are represented by the intensity of the emitter array and a matrix whose components are represented by the transmission coefficients of the pixels in the two-dimensional (2-D) spatial light modulator (SLM). Lens 1 is symbolic of a cylindrical lens system that distributes each of the vector's components along the columns of the matrix. The multiplication is performed componentwise as the light from the linear array is differently attenuated through each of the SLM pixel cells. The second cylindrical lens system (Lens 2) collects the exiting light along the row direction and integrates the sums on an array of detectors oriented perpendicularly to the emitter array. The output of the detectors is the resulting matrix–vector product. In this system the components of the vector and the matrix can be either binary or analog; the resulting accuracy in both cases depends on the specific devices used.

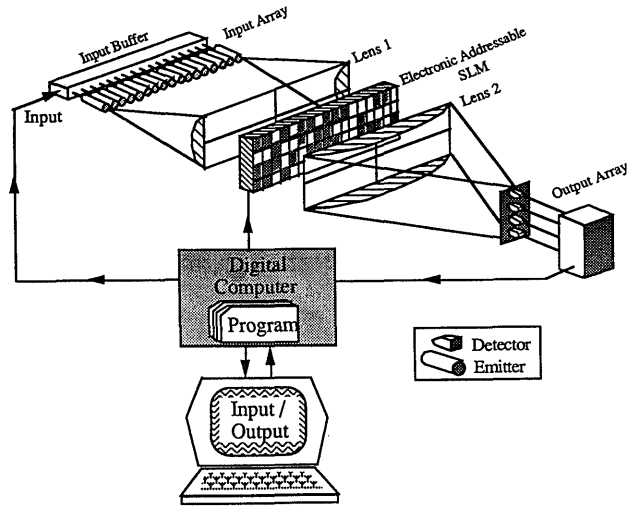

Fig. 2. One possible way to use an optical matrix–vector multiplier as a coprocessor to compute several of the parallel operations required for the ART1 algorithm.

As is also shown in Fig. 2, the OMVMP is typically controlled by a digital computer, which contains in its memory images of the vector and the matrix to be multiplied, as well as the controlling program. The vector and the matrix are loaded into the optical processor through electronic interface devices, and the resulting product is read back into memory from a controller for the detector array. In a sense this configuration uses the optical processor as a coprocessor to a digital computer.

To see how this system can be used as a coprocessor in the execution of the ART1 algorithm, refer to Fig. 3. In the figure it is assumed that the input pattern $\mathbf{I}$ is a 2-D binary image, and one possible way that it can be mapped onto the one-dimensional (1-D) emitter array is shown. We also see how the first memory template $\mathbf{T}_1$ is mapped onto the 2-D SLM device. With this assignment, the OMVMP will compute the dot products $\{(\mathbf{I} \cdot \mathbf{T}_k),\ 1 \le k \le n_c\}$. These dot products are read from the detector into memory, and the remainder of the algorithm is executed within the digital computer. This is a mode of operation similar to that used by Wunsch *et al.*[11] for the $4F$ optical correlator implementation of an ART1 network.

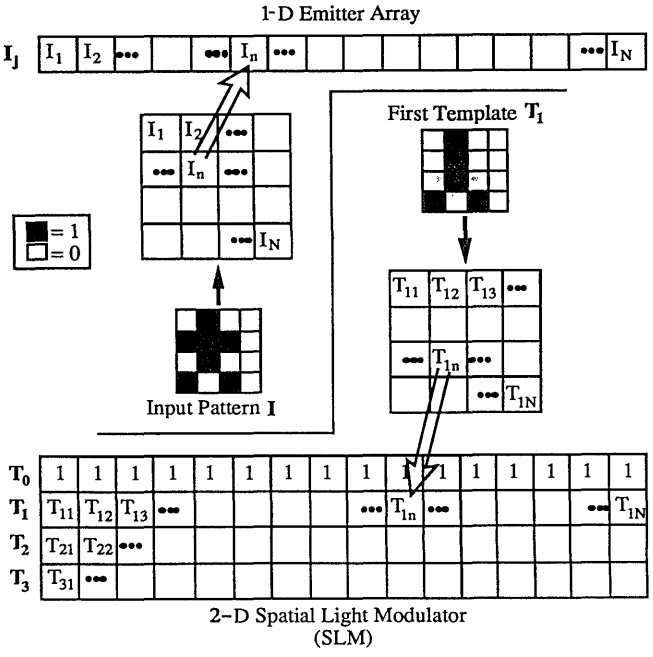

Fig. 3. How binary images and templates can be loaded into the optical processor arrays. Note that the $\mathbf{T}_0$ all-transmission row is used to compute the input vector norms. $\mathbf{T}_0$–$\mathbf{T}_3$, memory templates; $\mathbf{I}_j$, input pattern.

A simple change in the use of the basic OMVMP permits this processor to compute the vector norms $|\mathbf{I}|$ and $\{|\mathbf{T}_k|,\ 1 \le k \le n_c\}$ required in the ART1 algorithm. Figure 3 shows how $|\mathbf{I}|$ can be computed by providing an extra row on top of the 2-D SLM that is fully transmissive. This is equivalent to loading the unit vector $\mathbf{1}$ into the extra row on the SLM. The first detector in the focal plane of Lens 2 then integrates $(\mathbf{I} \cdot \mathbf{1})$, which is identical to $|\mathbf{I}|$. To compute the norms of the templates, we must take a second sequential step. The input emitter array must be set to the unit vector. With this vector on the input, the detectors then integrate $(\mathbf{1} \cdot \mathbf{T}_k)$, which is again identically equal to the norms $\{|\mathbf{T}_k|,\ 1 \le k \le n_c\}$.

Thus in two steps, most of the parallel operations listed under Optics in Table 1 can be computed by this coprocessor. The remaining operations are (1) new template copy and (2) existing template update. Since in the OMVMP the input vector and the templates must also be stored in the computer's memory, both of these operations could be performed in software by a serial computer. Amdahl's law,[15] which states that the maximum speedup possible through parallelism in an algorithm is set by the remaining serial operations, advises us to seek parallel implementations for these remaining ART1 operations. Subsection 3.B describes further modifications to the OMVMP that make these calculations possible within an optoelectronic processor.

B. Augmented Optical Matrix–Vector Processor

Several major modifications to the basic components of the OMVMP are shown in Fig. 4. With the addition of VLSI analog processing, these components become relabeled as (1) a smart-input array (SI array), (2) a smart SLM array (SSLM), and (3) a smart-output array (SO array). These new devices permit the copy and update operations to be performed optoelectronically and thus eliminate the need for a digital computer. Let us consider each modification separately.

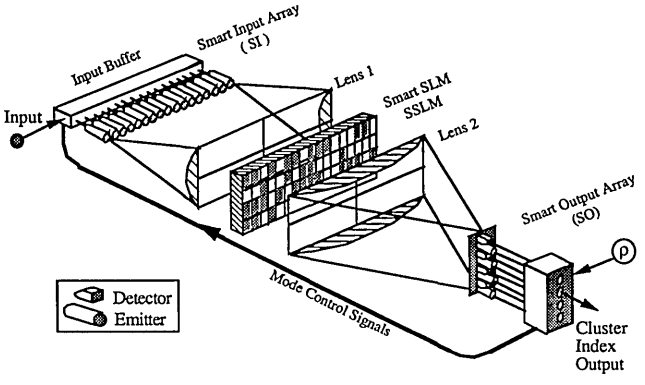

Fig. 4. Schematic architecture that uses optics to compute all the operations checked for optics in Table 1. The remainder of the computation is performed in analog electronics. Note the use of a smart SLM that is controlled by light signals in the forward and reverse directions. The mode control signals pathway is the only nonoptical communications between chips and could be implemented as a small cable. The output of the system is the index of the resonating cluster.

The SO array, shown in Fig. 5, replaces most of the functionality of the digital computer by a single VLSI CMOS chip. This chip incorporates a linear array of detectors, a linear array of emitters, an array of analog storage registers, analog scalar arithmetic operators, and a winner-take-all network. The latter implements the arg max function required in the algorithm. The registers store the input norm $|\mathbf{I}|$ computed through the first transparent row in the SLM and the template norms $|\mathbf{T}_k|$ that are computed by loading the unit vector on the SI array. These values are used at various stages in the ART1 algorithm and are combined with the dot products by using analog scalar arithmetic operators to feed the winner-take-all network, which is also implemented in analog circuits. Specialized control circuitry regulates the reset and search processes and controls the logic of the comparison operations. In addition, a single mode control signal, discussed below, is sent back to the SI array along a small cable.

Control of the SSLM is performed by the SO array indirectly through modulation of its emitters B_k, which shine light back through Lens 2 to illuminate a row on the modulator. Figure 6 shows a diagram of a cell on the SSLM. The reverse illumination helps perform the conjunction of the input pattern and the winning template, and it helps the copying of an input pattern into a new template. These operations occur in the cells of the SSLM when the reverse illumination highlights a template row and the SI array is in the appropriate mode of operation.

A blowup diagram of the SI array is given in Fig. 7, in which we see the four modes of operation con-

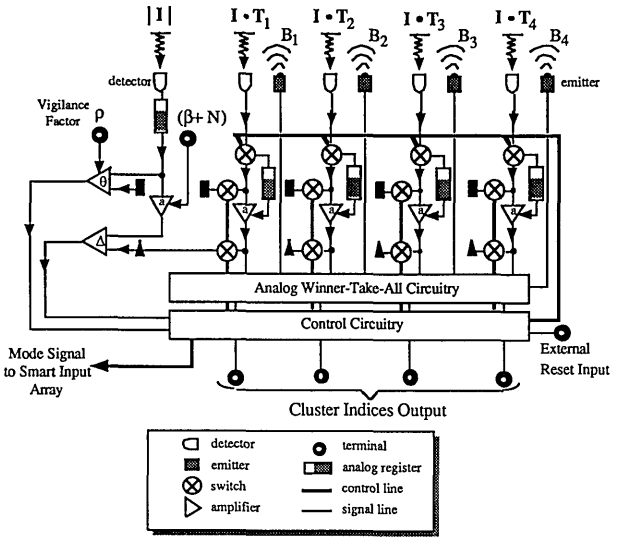

Fig. 5. Block diagram of the smart-output (SO) array chip design that would replace the digital computer and control the optical processor. Since this device simulates a winner-take-all neural network, only one output is on at a time. If the output is coded in binary, for example, then the number of pins on the chip package will not limit the maximum number of clusters. B_1–B_4, emitter outputs; 0, threshold.

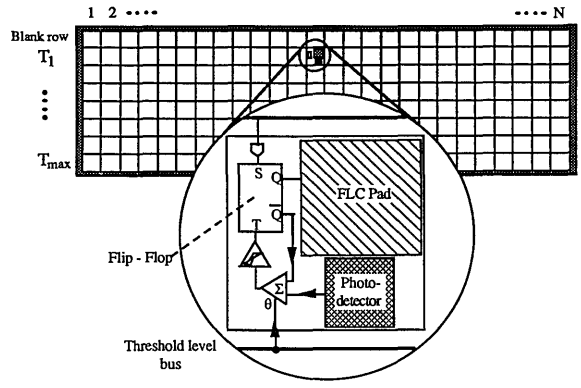

Fig. 6. Close-up of a reflective smart pixel on the SSLM implemented with ferroelectric liquid crystals (FLC's). Note that the flip-flop memory element has a set ON power-up circuit and that a threshold θ is globally bussed to each pixel. The summing circuit combines and thresholds the optical signals from the SI array and the SO array, along with an electrical signal representing the state of the flip flop. If the threshold is exceeded, a single clock pulse toggles the state.

trolled by the signal from the SO array. During the computation of the input-pattern-template dot products and the norm of the input pattern, the SI array is operated in mode 0. In this mode the emitter array illuminates the columns of the SSLM with the binary-input pattern previously clocked into the array's local buffers by means of a rapid shift register action. When operated in mode 3, all emitters are turned on to compute en masse the norms of the templates. Note that the SO array is not reverse illuminating the SSLM during these two modes of operation.

During the copy and update operations, the SO array will have selected a template row on the SSLM for modification, owing to either a match with an existing template or a novel pattern detection. This selection manifests itself through the reverse illumination of a template row on the SSLM. Referring again to Fig. 6, we see that the state (i.e., pass/on or block/off) of each pixel on the SSLM is determined by the state of a local flip-flop memory element. At power-up time, each state is set to logical ON by a

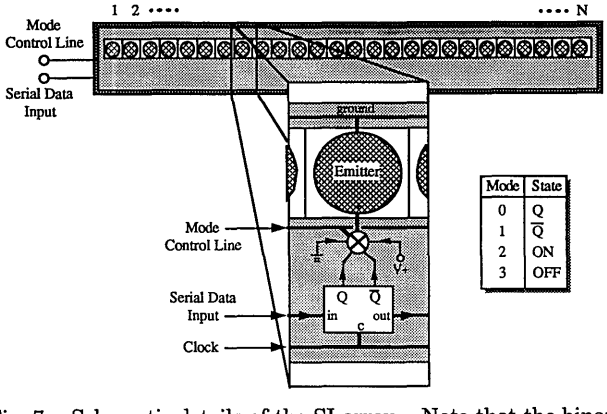

Fig. 7. Schematic details of the SI array. Note that the binary input pattern is clocked through a shift register to reach each emitter pixel. The four modes of operation are given in the inset and are controlled by the mode control lines.

reset circuit. The state of the pixel is toggled when a threshold amount of light is received on the pixel's photodetector. The threshold is adjusted so that no state change occurs when only the forward or the reverse illumination is present alone. Both must be simultaneously active before a toggle of a SSLM cell state can occur.

To copy an input pattern into a row on the SSLM, the SO array illuminates the correct row, while the SI array in mode 1 illuminates the columns to be toggled with the negation of the input pattern. Since the initial state of each pixel in an unused row is ON, the template pixels in the selected row and selected columns are toggled to the OFF state if the template pixel was initially ON and the input pixel OFF. A truth table of the state transitions is shown in Table 2. To perform the conjunctive update operation, the SO array illuminates the correct row, while the SI array, again in mode 1, illuminates the columns to be toggled with the negation of the input pattern. This operation is illustrated in Table 3, which shows the performance of an in-place conjunction of the template and the input pattern. The tables also show that the binary information stored on the SSLM does not change when the reverse illumination is removed, as indicated by the rows with $B_k = 0$.

The designs of these three devices, combined with the two cylindrical lens systems, constitute a processor that could efficiently implement the entire ART1 algorithm. Table 4 reaffirms this by presenting the steps in the algorithm that would be used in the optoelectronic processor. Note that the copy and update operations differ only in the cluster index used to backilluminate the SSLM. Section 4 presents the concept of a physical implementation of this system by using ferroelectric liquid-crystal spatial light modulators (FLC-SLM's).

4. Design for Optoelectronic Implementation

FLC's act to rotate the plane of polarization of transiting light. The degree of rotation depends on the strength of the longitudinal electric field and the length of the path through the material. The correct combination of thickness and field strength can produce a switchable mirror or a SLM. Devices now

Table 2. Truth Table Showing the Copying Operation Required to Generate a New Cluster Template[a]

			$\mathbf{T}_k \leftarrow \mathbf{I}$		
B_k	$\mathbf{I}_n$	$\bar{\mathbf{I}}_n$[b]	$\mathbf{T}_{kn}^{\text{initial}}$	Σ_n[c]	$\mathbf{T}_{kn}^{\text{final}}$
1	1	0	1	2	1
1	0	1	1[d]	3[d]	0[d]
0	1	0	1	1	1
0	0	1	1	2	1

[a] B_k is the output of the kth emitter on the SO array, and is ON when the kth template is to be created.

[b] SI array mode = 1, inverted input pattern.

[c] Threshold for toggle of SSLM pixel flip-flip ≅ 2.5 and $\Sigma_n = B_k + \bar{\mathbf{I}}_n + \mathbf{T}_{kn}^{\text{initial}}$.

[d] Values for the only condition in which a change of state of a template pixel occurs.

Table 3. Truth Table Showing the Operation Required to Update a Cluster Template[a]

$\mathbf{T}_k \leftarrow \mathbf{T}_k \cap \mathbf{I}$					
B_k	$\mathbf{I}_n$	$\bar{\mathbf{I}}_n$[b]	$\mathbf{T}_{kn}^{\text{initial}}$	Σ_n[c]	$\mathbf{T}_{kn}^{\text{final}}$
1	1	0	1	2	1
1	0	1	1[d]	3[d]	0[d]
1	1	0	0	1	0
1	0	1	0	2	0
0	1	0	1	1	1
0	0	1	1	2	1
0	1	0	0	0	0
0	0	1	0	1	0

[a] B_k is the output of the kth emitter on the SO array, and is ON when the kth template is to be updated.
[b] SI array mode = 1, inverted input pattern.
[c] Threshold for toggle of SSLM pixel flip-flip $\cong 2.5$ and $\Sigma_n = B_k + \bar{\mathbf{I}}_n + \mathbf{T}_{kn}^{\text{initial}}$.
[d] Values for the only condition in which a change of state of a template pixel occurs.

exist that integrate FLC-SLM's on the surface of CMOS VLSI integrated circuits.[16–19] When combined with silicon photodetectors, this class of device has the potential to implement the active components in the aforementioned architecture.

Figure 8 demonstrates how three FLC-SLM hybrid devices could perform the functions of the SI, SO, and SSLM arrays. The light source floods the FLC-SLM's on the SI and SO arrays with polarized light. This way the devices may act as emitters by reflecting the light toward the SSLM while modulating the polarization. Polarizing beam splitters pass rotated polarization beams and deflect the unrotated beams back toward the light source. A third polarizing beam splitter in front of the SSLM combines the polarized beams from the two emitting arrays, which permits threshold logic to be performed during the template copy and update operations.

This optical system does not require a coherent light source since no interference phenomenon is used in the processing. A narrow wavelength band may be required to achieve reasonable contrast ratios with the FLC-SLM's. Consequently, high-power light-emitting diode arrays may supply the input illumination.

This processor can potentially occupy a small volume. If the lenses are replaced with solid gradient-index elements, the implementation concept shown in Fig. 8 could occupy a volume of less than 10 cm^3. Alignment would be frozen by cementing the elements together into a solid structure. No computers would be required for operation, although external circuitry would be necessary to clock in the input array and to make use of the outputs.

Table 4. Mapping of the ART1 Algorithm Into Processing Steps of the Optical Processor

Step	SO Array	SI Array	SSLM
		SEARCH AND MATCH operation	
1.	Output mode = 0 to SI array	Shift in input, mode = 0	Subthreshold
2.	Integrate detectors	Emitters ON	
3.	Store input norm		
4.	Normalize dot products with template norms		
5.	If no active clusters, then GOTO COPY	Mode changing	Superthreshold, pixel toggling
6.	Winner-take-all finds max index		
7.	If FAIL first inequality, then GOTO COPY	Mode changing	Superthreshold, pixel toggling
8.	If FAIL second inequality, proceed as follows:		
	(1) Decrement number of active clusters		
	(2) If none left, then GOTO COPY	Mode changing	Superthreshold, pixel toggling
	(3) Otherwise, inhibit winning cluster and GOTO step 5		
9.	Resonance, output cluster		
10.	GOTO UPDATE	Mode changing	Superthreshold, pixel toggling
11.	GOTO step 1		
		COPY operation	
a.	Output mode = 1 to SI array	Invert input, mode = 1	Subthreshold
b.	Backemitter ON for next available cluster index	Emitters ON	Pixels toggle
c.	Backemitter OFF		
d.	Output mode = 2 to SI array	All ones, mode = 2	Subthreshold
e.	Integrate detectors	Emitters ON	
f.	Store template norms		
g.	GOTO step 1		
		UPDATE operation	
a.	Output mode = 1 to SI array	Invert input, mode = 1	Subthreshold
b.	Backemitter ON for winning cluster index	Emitters ON	Pixels toggle
c.	Backemitter OFF		
d.	Output mode = 2 to SI array	All 1's, mode = 2	Subthreshold
e.	Integrate detectors	Emitters ON	
f.	Store template norms		
g.	GOTO step 1		

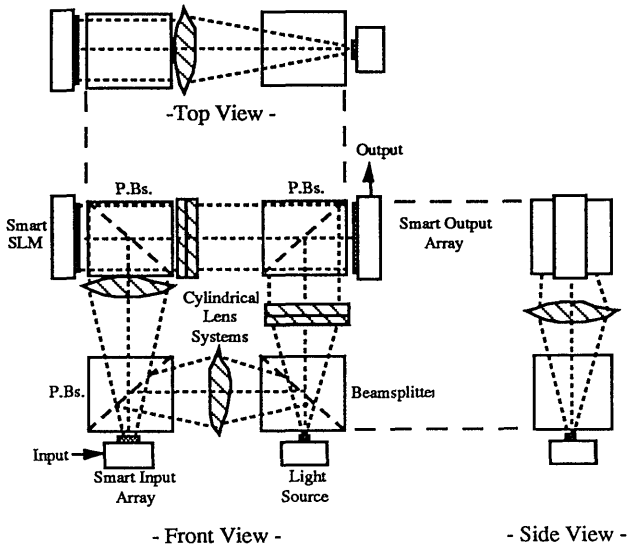

Fig. 8. One possible physical implementation of an ART1 neural network by using FLC-SLM CMOS devices. P.Bs., polarizing beam splitter.

The approximate temporal performance of this implementation concept may be calculated. From Table 1 we conclude that the dominant number of operations is generated by the parallel dot products in row 3 and by the template norms in row 4. With the assumption that one 8-bit MULTIPLY operation is equivalent to approximately ten ADD operations, we can make an order-of-magnitude estimate of the number of operations necessary to process one input pattern: $N_{ops} \approx 10\, M_{templates} N_{bits}$, where $M_{templates}$ is the maximum number of possible templates and N_{bits} is the length of the input vector. For example, if $M_{templates} = 100$ and $N_{bits} = 1024$, then the number of operations is approximately 1×10^6. This implies that a standard Sun SPARCStation 2 could process ~25 input vectors/s.

The approximate cycle time for the processor can also be calculated. Upon analysis of the steps in Table 4, we see that the cycle time is dominated by the integration times of the detectors in the SO array. In fact, the cycle time can be estimated to be $T_{cycle} \approx 2T_{integration}$. This is derived from the fact that each input pattern requires two parallel detector integrations. The integration time $T_{integration}$ strongly depends on the specific properties of the detectors, the time constants of the analog circuits, the minimum signal-to-noise ratio required by the algorithm, and the specific application in which the network is used. Since the total signal level is controlled by the intensity of the illumination source, a great deal of latitude exists in estimating the minimum permittable integration times. A value in the range 10^{-5}–10^{-4} s is not unreasonable for currently available detectors and light sources. Using this range of times with the example sizes of $M_{templates} = 100$ and $N_{bits} = 1024$, we calculate the approximate number of operations per second for this optical processor to be $R_{op/s} \approx 10^{10}$ to 10^{11}. Although this estimate is based on a number of simplifying assumptions, it implies that this processor could potentially cluster upwards of $10^{11}/10^6 = 10^5$ input images/s of size 1024 binary pixels.

Operational errors in classification occur in this system when either fixed-pattern noise or nonuniform illumination is too large. To begin to study the effects of these on system performance, consider a Monte Carlo simulation that was conducted on a digital computer. Figure 9 shows the frequency of error caused by uniform additive random fixed-pattern noise on the SI array, the SSLM, and the SO array. An error is defined as the case in which an input pattern is placed in a wrong cluster. Each curve in the figure represents the average behavior of a large number of simulations with random input vectors of a fixed size, in which the range of random variation is systematically increased. For each of these simulations, two templates are matched against the input pattern. The templates and the input pattern all have the same norms. In truth, the first template exactly matches the input, while the second template differs from the first by a preset number of bits. The difference between the templates is varied to construct the family of smooth curves shown in the figure (each curve is labeled with the Hamming distance between templates). These simulation results show that ART1 requires approximately ±1% uniformity for no errors to occur in the hardest discrimination case (a template Hamming distance of 1 out of 1024 bits). The number of errors grows slowly as the size of the nonuniformity increases. The system behaves more robustly for clusters that are better separated, as indicated by larger Hamming distances. It is difficult to generalize these results to cases with larger number of clusters owing to the nature of the ART1 similarity measures.

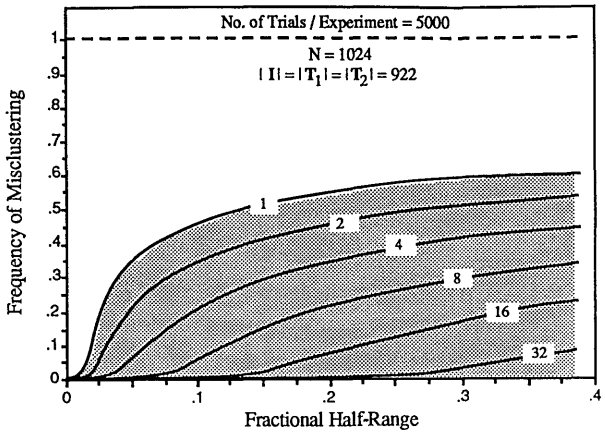

Fig. 9. Results of sets of Monte Carlo simulations, each computing the frequency of misclustering as a function of the fractional half-range of additive-fixed-pattern noise on the SI array and the SSLM. The noise was picked from a uniform random distribution over the indicated range. A random input pattern of fixed size is submitted to an ART1 module that is preset with two cluster templates, one being identical to the input. The templates are also of fixed size equal to that of the input pattern. The curves are labeled with the Hamming distance between the two competing cluster templates.

Figure 10 shows the results of a second set of Monte Carlo simulations that explore the sensitivity of the template update operation to uniform additive random fixed-pattern noise. A large number of simulations were performed with an increasing range of variation in the noise, which produced the smooth curve. The vertical axis gives the average number of bit errors at each variance. A bit error is counted when an incorrect toggle occurs or when a correct toggle does not occur. No errors appear until the half-width of the variation exceeds 20%. If the system is designed with nonuniformities compatible with low classification error, then update errors will not occur.

5. Discussion

A design for an optical implementation of the ART1 neural network has been presented. This design has the advantages of a totally optoelectronic implementation: optical interconnects (fan-in and fan-out), VLSI analog processing, low power consumption, and compact monolithic packaging. No software would be required for the control of the algorithm. The input pattern is clocked into the input buffer, and after a delay of approximately $2T_{\text{integration}}$, the classification code appears on the output of the SO array. The design has the potential to classify 10^5 small images/s into many tens of classes, while remaining resilient to new or novel input patterns. The technology to build the components of this design are within the reach of today's technology.

The major disadvantage of this design is in the mapping of a 2-D image onto the 1-D input buffer array. In many applications, input images will be larger than the 32 × 32 pixel image array used in the example of Section 4. Typical images of 256 × 256 pixels are not unusual in the applications of ART1 mentioned in the introduction. This size image would require a SLM with 64,000 pixels along one axis. With pixel sizes of 10 μm × 10 μm, a slab of silicon 65 cm long would be required, clearly not within the reach of wafer-scale circuits.

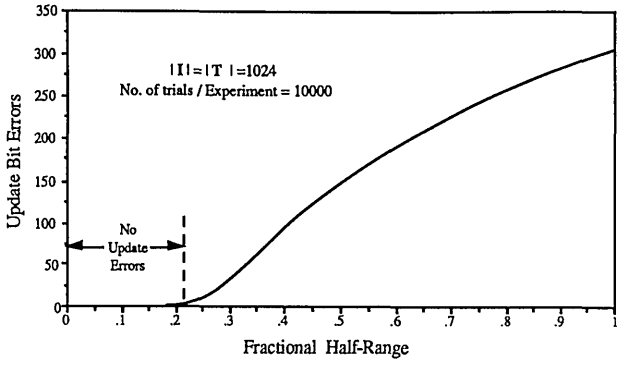

Fig. 10. Results of a set of Monte Carlo simulations that computes the frequency of toggle errors during a template update operation as a function of the fractional half-range of additive fixed-pattern noise on the SSLM and the SO array. The noise was picked from a uniform random distribution over the indicated range. The vertical axis is the average number of bit errors.

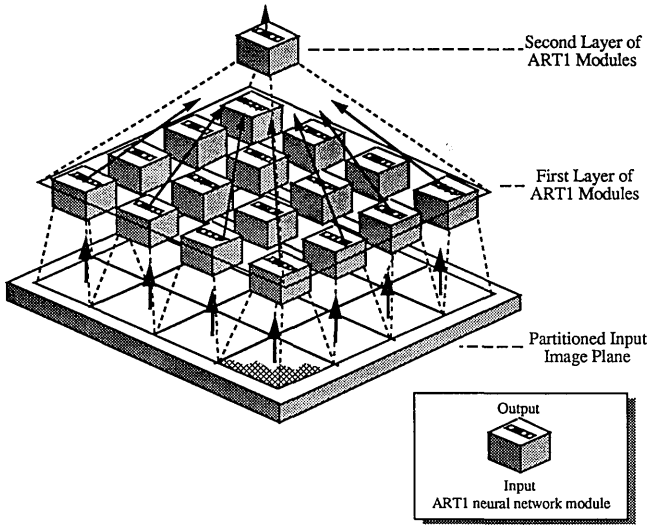

Fig. 11. Illustration of a simple macrocircuit that combines ART1 modules into a hierarchy.

As demonstrated by the research of Healy and Caudell,[20] the smaller size of the input image is not necessarily a limitation. Their research shows that macrocircuits formed from combinations of ART1 modules can be designed to compute complex logical functions of the input pixel images. The simplest example of a macrocircuit is a hierarchy of ART1 modules. An example of this hierarchical structure is shown in Fig. 11 for a 128 × 128 pixel image. The image is partitioned into small nonoverlapping patches, each patch being fed to individual 32 × 32 ART1 modules. The outputs of this layer of separate ART1 modules are then funneled into a single ART1 module that combines the lower level results. The vigilance parameters are adjusted so that this architecture acts as a voting machine, with the higher ART1 module producing an output code for the most

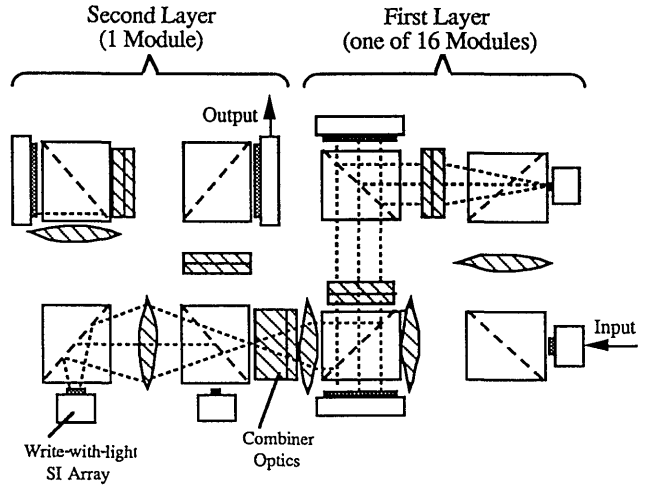

Fig. 12. Optical processors cascaded together to form a macrocircuit of ART1 neural networks. Note that the SI array has been modified to include detectors that permit optical writing of the input vector. This extra circuitry also provides a mechanism for reading the learned templates stored on the SSLM.

popular interpretation of the input object. More complicated functionality has been designed into network macrocircuits.[21]

Figure 12 shows how this architecture can be implemented with the optoelectronic processor. The temporal performance of such a hierarchical implementation is reduced by a factor proportional to the number of layers. A modification of the SI array design is necessary to optically cascade these modules. In addition to the electronic serial loading of the input vector, an optical parallel load, or write-with-light, mode is required. At each pixel in this linear array, a photodetector is added that can be clocked to read the pattern of light on the array and to load this information into the input buffers. Additional optics are necessary to map the square subimage onto a linear vector and to arrange the outputs of the first layer for input to the second layer. This modular approach overcomes the main size limitation of these devices. Macrocircuits for practical size applications can be constructed from these building blocks.

Conclusion

Adaptive resonance theory neural networks are becoming an important component of neural network industrial applications. Electronic implementation of these network models has proven difficult to scale to large-input dimensionality. Here a new implementation of ART1 has been proposed that efficiently combines optical and electronic devices. All parallel computations were performed by the optics, while serial operations were performed in electronics. One possible physical implementation of this architecture has been proposed. Macrocircuits can be constructed from these modules to perform complex logical functions. Based on the write-with-light modification, this system can also be modified to permit readout and readin of the learned templates on the SSLM, which permits mass production of trained systems.

The author acknowledges the helpful suggestions of Karel Zikan, John Bell, Donald Wunsch II, David Capps, Kris Johnson, Kelvin Wagner, Janusz Kowalik, and the journal reviewers.

References

1. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," in *Computer Vision, Graphics, and Image Processing,* L. Shapiro, N. Badles, H. Freeman, T. Huang, and A. Rosenfield, eds. (Academic, New York, 1987), Vol. 37, pp. 54–115.
2. B. Moore, "ART1 and pattern clustering," in *Proceedings of the 1988 Connectionist Summer School,* D. Touretzky and G. Hinton, eds. (Carnegie U. Press, Pittsburgh, Pa., 1989), pp. 174–185.
3. G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system," Neural Networks **4,** 759–772 (1991).
4. M. Georgiopoulos, G. L. Heileman, and J. Huang, "Properties of learning related to pattern diversity in ART1," Neural Networks **4,** 751–758 (1991).
5. T. P. Caudell, S. Smith, R. Escobedo, and C. Johnson, "A neural database system for reusable design," in *Proceedings of the International Joint Conference on Neural Networks,* C. L. Teo, ed. (Institute of Electrical and Electronics Engineers, New York, 1991), Vol. 1, pp. 254–261.
6. A. A. Baloch and A. M. Waxman, "A neural system for behavioral conditioning of mobile robots," in *Proceedings of the International Joint Conference on Neural Networks,* W. Freeman and B. Kosko, eds. (Institute of Electrical and Electronics Engineers, New York, 1990), Vol. II, pp. 723–728.
7. V. G. Kaburlasos, D. D. Egbert, and M. Rao, "Hardware implementation of the adaptive resonance theory neural network," in *Proceedings of the First Golden West International Conference on Intelligent Systems,* C. G. Looney, ed. (University of Nevada, Reno, Nev., 1991), Vol. 1, pp. 21–28.
8. S. W. Tsay and R. W. Newcomb, "VLSI implementation of ART1 memories," IEEE Trans. Neural Networks **2,** 214–221 (1991).
9. D. Psaltis, N. H. Farhat, A. Prata, and E. Peak, "Optical implementation of the Hopfield model," Appl. Opt. **24,** 1469–1475 (1985).
10. H. J. Caulfield and D. Armitage, "Adaptive resonance theory of optical pattern recognition," Appl. Opt. **28,** 4060–4061 (1989).
11. D. Wunsch, T. P. Caudell, D. Capps, and A. Falk, "Optoelectronic learning machine," in *OSA Annual Meeting,* Vol. 15 of 1990 OSA Technical Digest Series (Optical Society of America, Washington, D.C., 1990), paper MJJ5.
12. A. B. VanderLugt, "Signal detection by complex spatial filtering," IEEE Trans. Inf. Theory **IT-10,** 139–145 (1964).
13. D. O. Hebb, *The Organization of Behavior* (Wiley, New York, 1949), Introd. and Chap. 4.
14. N. H. Farhat and D. Psaltis, "Optical implementation of associative memory based on models of neural networks," in *Optical Signal Processing,* J. Horner, ed. (Academic, New York, 1987), pp. 129–161.
15. G. Amdahl, "The validity of the single processor approach to achieving large scale computing capabilities," AFIPS Nat. Comput. Conf. Expo. Conf. Proc. **30,** 483–485 (1967).
16. M. G. Robinson, L. Zhang, K. M. Johnson, and D. A. Jared, "Custom electro-optical devices for optically implemented neuromorphic computing systems," in *OSA Annual Meeting,* Vol. 15 of 1990 OSA Technical Digest Series (Optical Society of America, Washington, D.C., 1990), paper MVV9.
17. K. M. Johnson, M. A. Handschy, and L. A. Pagano-Stauffer, "Optical computing and image processing with ferroelectric liquid crystals," Opt. Eng. **25,** 385–391 (1987).
18. L. K. Cotter, T. J. Drabik, R. J. Dillion, and M. A. Handschy, "Ferroelectric-liquid-crystal/silicon-integrated-circuit spatial light modulator," Opt. Lett. **15,** 291–293 (1990).
19. M. A. Handschy, T. J. Drabik, L. K. Cotter, and S. D. Gaalema, "Fast ferroelectric-liquid-crystal spatial light modulator with silicon-integrated-circuit active backplane," in *Optical and Digital Gallium Arsenide Technologies for Signal Processing Applications,* M. P. Bendett, D. H. Butler, A. Prabhokar, and A. Yang, eds., Proc. Soc. Photo-Opt. Instrum. Eng. **1291,** 158–164 (1990).
20. M. J. Healy and T. P. Caudell, "Application of ART1 hierarchies to recognition of complex objects," in *Conference on Neural Networks for Automatic Target Recognition,* S. Grossberg, ed. (Wang Institute, Boston University, Boston, Mass., 1991), p. 29.
21. M. J. Healy, "A logical architecture for supervised learning," in *Proceedings of the International Joint Conference on Neural Networks,* C. L. Teo, ed. (Institute of Electrical and Electronics Engineers, New York, 1991), Vol. 1, pp. 190–195.