

A Novel Associative Memory Implemented Using Collective Computation

Massimo Sivilotti, Michael Emerling, and Carver Mead

Department of Computer Science
California Institute of Technology, Pasadena

Abstract

A radically new type of associative memory, the ASSOCMEM, has been implemented in VLSI and tested. Analog circuit techniques are used to construct a network that evolves towards fully restored (digital) fixed-points that are the memories of the system. Association occurs on the whole source word, each bit of which may assume a continuous analog value. The network does not require the distinction of a search *key* from a *data* field in either the source or target words. A *key* may be dynamically defined by differentially weighting any subset of the source word. The key need not be exact; the system will evolve to the closest memory. In the case when the key is the whole input word, the system may be thought of as performing error correction.

1 Introduction

Conventional digital associative memories [1,2,3] have two characteristic shortcomings: (i) the association occurs on an a priori defined key, which must be supplied exactly, (ii) resolution of multiple matches often demands an additional serial polling algorithm (slow). In addition, they only work on digital (restored) data, and thus are useless for recognition of patterns with grey scale. We have designed and tested a radically new genre of associative memory. In the ASSOCMEM, association takes place on the word as a whole; the result is the word stored in memory that is *closest* to the supplied word. There is no distinction made between *key* and *data*. The convergence of the association process is guaranteed by theory. The length of time required to perform the association depends, in some sense, on the *distance* between the input and the result.

This novel behaviour is obtained through the use of analog circuit techniques to perform a "collective computation" [4]. Each bit of the word is represented by the state of an active element (amplifier). These elements are interconnected in a way to make the desired memory contents stable states of the network. An association consists of setting the initial conditions of the network to a neutral state, then letting the network evolve to a fixed-point.

A very simple example of such a network is the flip-flop illustrated in Fig. 1. It may be considered to operate in a continuous range between the signal rails $+1$ and -1 . The neutral state is the well-known "hung" condition where both inputs and outputs are equal. When the system is released, any difference between the inputs will cause the flip-flop to evolve to the closest stable state. This particular system may be thought of as having two memories, $(+1, -1)$ and $(-1, +1)$.

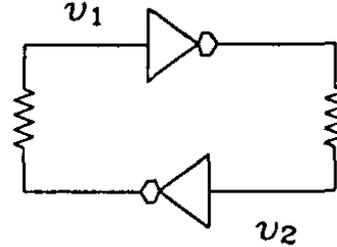
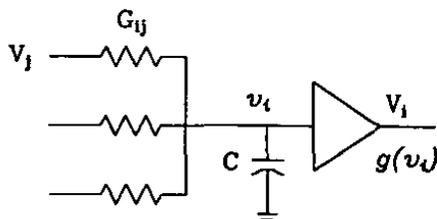


Figure 1: Simple Flip-Flop.

In general, a network will be represented by an interconnection matrix, as in Fig. 2. A considerable amount can be said about the behavioural properties of such a system: J. Hopfield has shown [5] that a symmetric interconnect matrix guarantees the existence of stable states (i.e. that the network will not oscillate, but will ultimately reach convergence), and has developed a synthetic technique for deriving a matrix with specified stable states [6]. This technique consists, in essence, of setting up a matrix such that every active element reinforces every other active element, when the system is in the stable state. The stable state vector is represented by \vec{v} , ($v_i \in \{-1, +1\}$), and the connection between the output of amplifier j and the input of amplifier i represented by T_{ij} . The connection matrix may be obtained by taking the outer product $\vec{v} \times \vec{v}$ (See Fig. 3). This matrix is guaranteed to have the desired symmetry property (i.e. $T_{ij} = T_{ji}$).



$$C \cdot \frac{dv_i}{dt} = \sum_{j=1}^N G_{ij} (\pm V_j - v_i)$$

$$= \sum_{j=1}^N G_{ij} (\pm g(v_j) - v_i)$$

Figure 2: Interconnection of active elements, and dynamic behaviour of system.

$$\vec{v} \times \vec{v} = \begin{bmatrix} +1 & -1 & -1 \\ -1 & +1 & +1 \\ -1 & +1 & +1 \end{bmatrix}$$

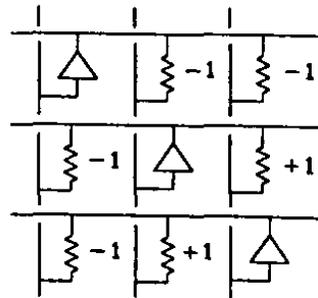


Figure 3: Reinforcement of stable state with matrix generated by outer product technique.

$$\vec{v} = (+1, -1, -1)$$

The evolution of the state of such a network with time is interpreted as an association in progress. Hopfield has proven the existence of an energy function [5] that the system minimizes. The local minima of this function constitute the fixed-points, or memories, of the network.

Such associative memories have several appealing features. First, it is possible to indicate *confidence* on a bit-by-bit basis, by setting the initial conditions appropriately, i.e. by making certain bits weigh more heavily, if desired. This property allows the ASSOCMEM to accept analog inputs, thus permitting it to process image or sound data. Secondly, all the bits are treated uniformly. The device may be thought of as an error-correcting machine, where randomly-occurring bit errors are corrected. Thirdly, the actual memories are *stored in the interconnect* in a highly redundant fashion, making this architecture naturally fault tolerant. Simulation results have indicated that 10% to 20% of all connections may be destroyed with practically no loss of functionality. If a small number of amplifiers are non-functional, they can still be "out-voted" by the rest of the network, and errors limited to those bits only.

2 The Chip: Design and Fabrication

The first version of the ASSOCMEM was designed in $4\mu\text{m}$ nMOS technology, and has a dynamically programmable full interconnect. Consequently, every amplifier must be connectable to every other amplifier, dictating a mesh topology with the amplifiers on the diagonal, and their input and output lines running orthogonal to each other. At each off-diagonal node, an interconnection "conductance", G_{ij} , is located. This topology (used in Fig. 3) produces a two-dimensional embedding of the interconnect graph implied by Fig. 2.

2.1 Design of the active elements

In order to achieve a general computation, both positive and negative signals and interconnect values must be representable. It is not possible to fabricate negative resistances, as would be implied by a negative matrix element. The solution was to design the amplifiers to take differential inputs and generate complementary outputs. These inputs may be thought of as excitatory and inhibitory (ie. non-inverting and inverting). This duality allows us to represent "negative" V_j 's while using only positive signals. Also, it guarantees symmetry between "positive" and "negative" V_j 's, since they are in fact the same voltage, merely on different lines (Fig. 4). The matched parameters of VLSI devices ensure a high CMRR for this configuration.

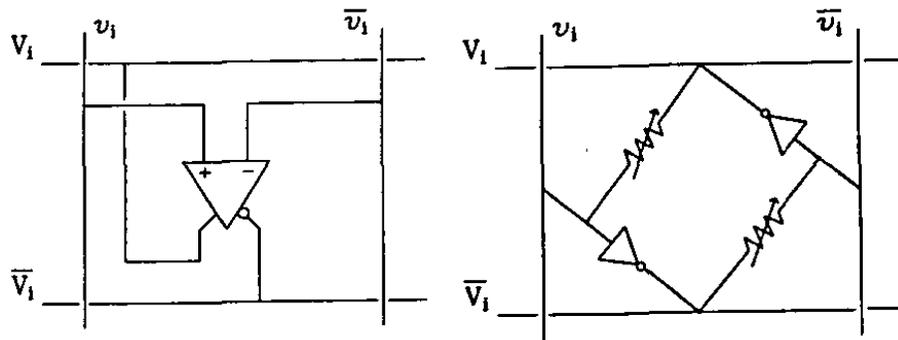


Figure 4a: Dual-rail signal representation Figure 4b: Variable Feedback in T_{ii}

This dual-rail voltage representation on both input and output to the active elements greatly simplifies their design. The only constraint on the transfer function $g(v_j)$ is monotonicity (to guarantee convergence, by the energy theorem), and that the gain be sufficiently large. A loop gain of greater than 1 is required to create distinct stable states; a high gain guarantees that all outputs will be driven to the rails. We chose to implement the amplifiers with the simplest circuit element possible: an nMOS depletion-load inverter. Two inverters are required for each amplifier: one between the non-inverting input and the complemented output, and one between the inverting input and the uncomplemented output. Note that feedback through the *matrix* is required to ensure that the amplifier outputs are indeed complementary. If the two inverters are replaced by a pair of cross-coupled NOR gates (i.e. a simple SR flip-flop), this complementarity can be enforced at each diagonal (T_{ii}) element.

The design that was implemented allowed for both modes of operation to be tested, by providing a variable strength cross-connection (CROSS_COUPLE) between the two inverters. The circuit diagram for the T_{ii} element is shown in Fig. 5. Two pass transistors, gated on ϕ_1 , allow the inputs of the amplifier to be latched. This capability is required for programming the interconnection matrix, and is desirable to halt the operation of the memory, for test purposes.

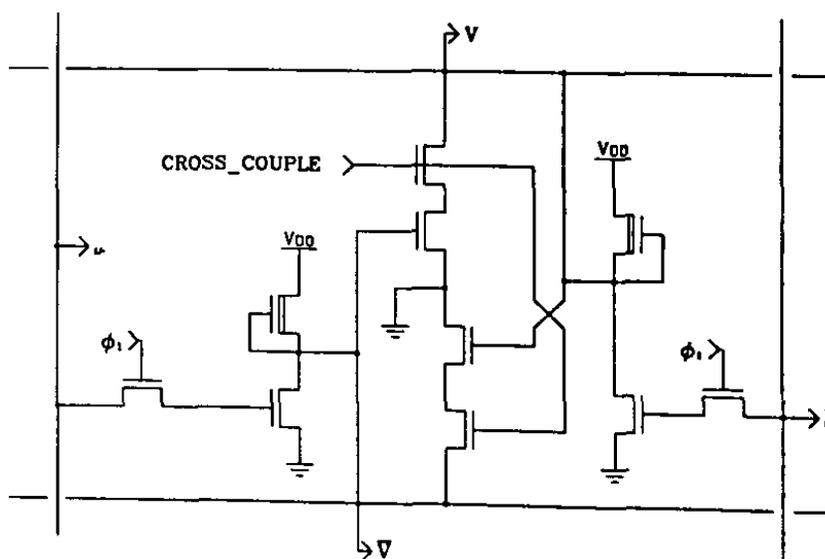


Figure 5: T_{ii} Element Schematic

2.2 Resistive Interconnect

The resistive coupling, representing the G_{ij} matrix, is provided by pass transistors, which constitute the functional part of the " G_{ij} Element" (Fig. 6), located at every off-diagonal location of the chip. These pass transistors are controlled by the tri-flop T_{ij} cell, which provides three interconnection strengths (+1, 0, -1). When $T_{ij} = +1$, the + output of amplifier j is connected to the + input of amplifier i (excitation), and the - output is connected to the - input of the respective amplifiers (negative inhibition \Rightarrow excitation); the opposite is true for $T_{ij} = -1$. If $T_{ij} = 0$, all transistors are disconnected, and amplifier j has no *direct* influence on amplifier i . Each pass transistor is in series with an ENABLE transistor that allows the matrix to be selectively disengaged, providing a "single-step" operation.

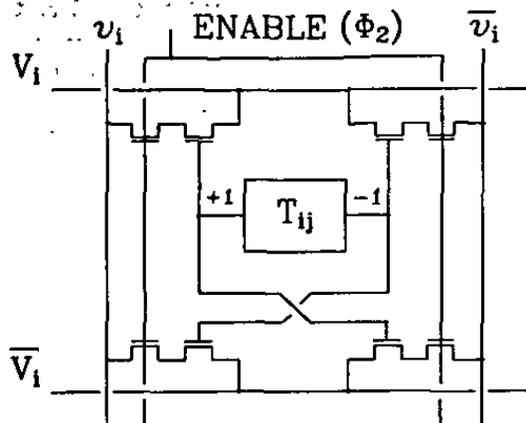


Figure 6: Resistive Interconnect

2.3 Programming the T_{ij} Elements

The associative memory can be programmed to have a particular vector as a stable state by adding the outer product of that vector with itself to the existing T matrix. In the ASSOCMEM, this operation is a primitive of the hardware, and has been generalized to allow taking outer products of any two vectors. The components of the outer product are produced in place at the T_{ij} element where they are required, by placing one vector on the horizontal signal lines, and the other on the vertical lines. This calculation requires only a 1-bit by 1-bit multiplication (for vectors containing +1 and -1 exclusively), which can be implemented by an AND structure (Fig. 7). The result of the correlation is added to the present contents of the T_{ij} cell, and the result is stored in the tri-flop T_{ij} latch.

The addition operation is truncated, and the behaviour is described in Table 1. It should be noted that this operation is *not* associative (the final matrix depends on the order in which the memories were added); however, the symmetry of the final matrix is guaranteed (since all intermediate sums comprised symmetric matrices, in the case of outer products taken of a vector with itself).

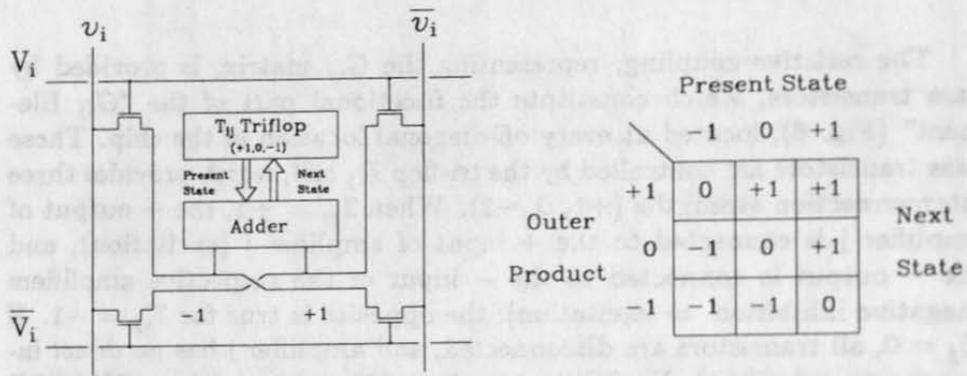


Figure 7 and Table 1: Programming the matrix

The effect of truncating the matrix in this fashion was one aspect of the theory we were interested in investigating. Software simulation of this scheme indicated that little functionality would be lost, in the case of uncorrelated memories (memories sufficiently far apart in Hamming space, so that no small number of bits become crucial in differentiating between them).

The complete schematic of the T_{ij} element, and the interconnection "resistors", is shown in Fig. 8. It should be pointed out that all but four of the transistors are required just to implement the *programmability* of the ASSOCMEM. In many applications, the interconnect matrix could be determined a priori, and the chip fabricated to this specification. We estimate that 100 times as many active elements could be fabricated within the same die size, for a predetermined function.

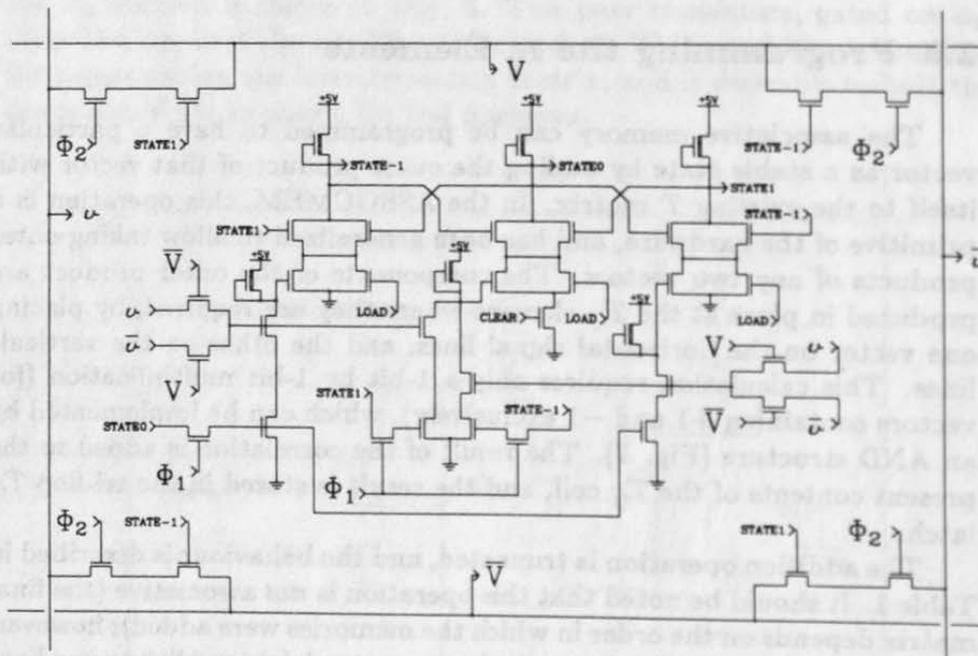


Figure 8: Schematic of T_{ij} element

3 Fabrication History

The first version of this chip was fabricated by MOSIS, the ARPA chip broker, on run M41U, in December 1983. It consisted of a 22×22 matrix, measuring $6700\mu\text{m} \times 5700\mu\text{m}$, and containing over 20,000 transistors (Fig. 9). The ASSOCMEM required 53 pads, as both inputs of each amplifier were brought out to pads. This redundancy was incorporated to increase flexibility in testing the chip. Only one input per amplifier is required, as the complement could be generated on-chip.

A second version of the chip was fabricated in May 1984, correcting some defects that made sections of the first chip non-functional and non-observable. This redesign yielded testable chips; however, the chips were returned unbonded, and only a few could be bonded locally and tested. The yield was low, and no single project was fully functional.

The desire for a larger sample size of chips prompted a third re-submission, which incorporated a set of minor modifications making it possible for MOSIS to package and bond the ASSOCMEM. This set of chips was returned in November 1984, and is currently being tested.

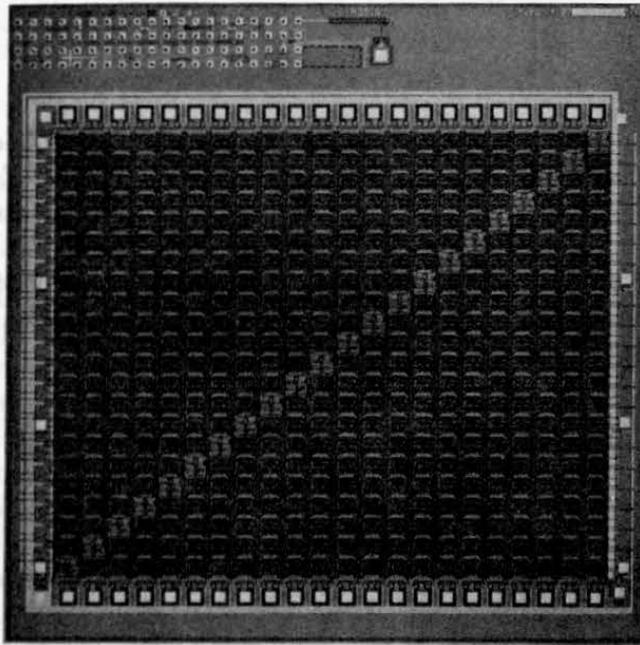


Figure 9: The ASSOCMEM

4 Experimental Results

The ASSOCMEM was tested on a dedicated, workstation-based functional tester. The first sets of experiments were designed to measure the input-output characteristics of the active elements. The input of a single T_{ii} element was connected, (via T_{ij} elements) to the outputs of the other amplifiers. The output of this element was monitored for 220,000 randomly chosen input vectors (this operation corresponds to taking the inner product of the random input vector with a vector that is all 1s).

The resulting output was plotted as a function of the arithmetic sum of the input bits (i.e. the number of + 1's in the input vector minus the number of - 1's), as shown in Fig. 10. The symmetry of the graph demonstrates that the output is unbiased and the input sum to the active element is balanced for an input vector containing equal numbers of + 1's and - 1's.

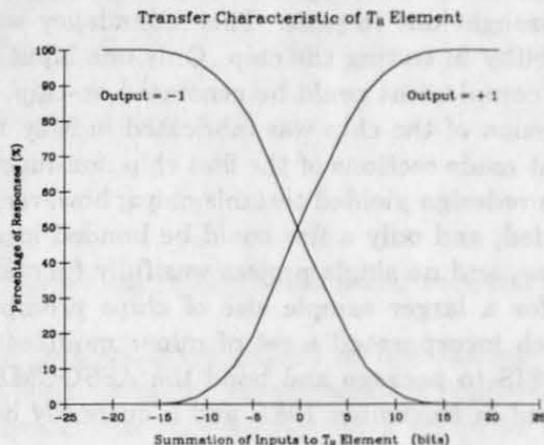


Figure 10: T_{ij} Transfer Characteristic

The ability to arbitrarily set a single T_{ij} element confirmed the functionality of the adder elements and tri-flops within each T_{ij} element. Several ring oscillators were programmed into the matrix, ranging from 3 stages to 19 stages. A typical waveform is shown in Fig. 11. The results in Fig. 12 indicate a characteristic propagation delay of 510ns per stage. The ring oscillators required an asymmetric connection matrix, as oscillatory behavior is excluded for symmetric matrices.

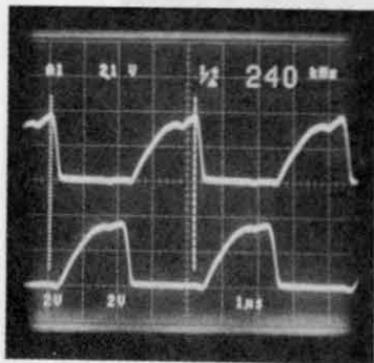


Figure 11: Ring Oscillator

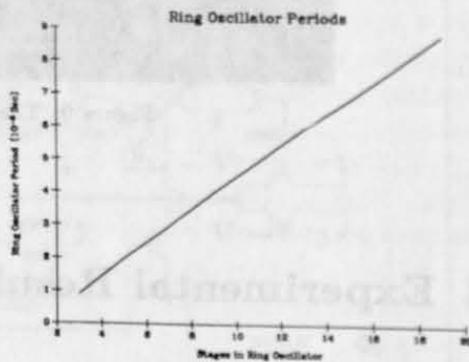


Figure 12: Ring Oscillator Periods

The next experiment was to perform an actual association. Two random memories were programmed into the matrix, via the on-chip outer product and adder mechanisms, and were observed to be, in fact, fixed points of the network. Due to the fast T_{ii} elements, the convergence of the network, including the driving of the output pads, was generally completed within a few microseconds.

If the function of the ASSOCMEM is considered to be that of an error-correcting code, it is useful to consider the final stable state as a function of the bit-errors in the initial state. Typical experimental results demonstrating the distinct regions of convergence for a two memory system are shown in Fig. 13. Effectively, this system always converges to the memory nearest in Hamming space to the initial condition.

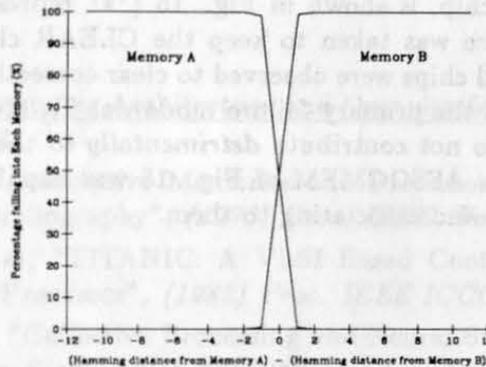


Figure 13: Convergence Properties

By using ϕ_1 and ϕ_2 as non-overlapping "clocks" to successively engage and disengage the connection matrix and T_{ii} elements, it is possible to "single-step" through the convergence operation. Each step corresponds to the network synchronously performing an inner product calculation to obtain the next state from the present state. It can be proven that a network which converges in synchronous-update mode will also do so asynchronously (i.e. ϕ_1 and ϕ_2 ON simultaneously), although the converse is *not* true. For networks that do converge in single-step mode, it is interesting to observe the process in action (see Fig. 14). In this illustration, note that the intermediate results are restored approximations to the analog voltages within the ASSOCMEM, and that although two iterations appear identical, the convergence is not yet complete.

Simulation results [6] have suggested an empirical limit on the number of uncorrelated memories that can be programmed into a Hopfield network to be approximately 15% of the number of active elements. Beyond this, spurious stable states begin to appear, often displacing desired memories. Since the network is based on mutual inhibition and reinforcement, any correlation between memories causes some bits to be weighted more heavily than others. Statistical orthogonality between memories minimizes this effect.

A second limitation on the number of stable states arises due to the clipped nature of the T_{ij} matrix. Hopfield and Feinstein [7] showed that the "attraction" of stable states decays exponentially with the number of memories subsequently added to the matrix.

For the ASSOCMEM, it was found that up to three stable states could be reliably programmed, along with their complements (which appear due to the symmetry of the system). At all stages of the experimentation, consistency with simulation results was observed.

Of the ten chips returned by MOSIS, all were found to be functional to the extent of permitting at least two stable states. A cif defect made a particular T_{ij} element malfunction in all chips. On some chips, sizeable numbers of T_{ij} elements were inoperative. A sample map, from a particularly bad chip, is shown in Fig. 15 ("*" represent good T_{ij} elements). Since care was taken to keep the CLEAR circuit simple, and the matrices on all chips were observed to clear correctly (and totally), it is postulated that the primary failure mode was T_{ij} stuck-at-zero faults. Since null T_{ij} 's do not contribute *detrimentally* to the operation of the network, even the ASSOCMEM of Fig. 15 was capable of memorizing two stable states, and associating to them.

```
Incremental Association:

(stable states (hex): OFOF0F, 00OFFF)
(vector written (hex): 00000F)

Iteration 0: V = 000000000000000001111
Iteration 1: V = 0000000000111100001111
Iteration 2: V = 0000000000111100001111
Iteration 3: V = 0001010000111100001111
Iteration 4: V = 0011110000111100001111
```

Figure 14: Single-Step Association

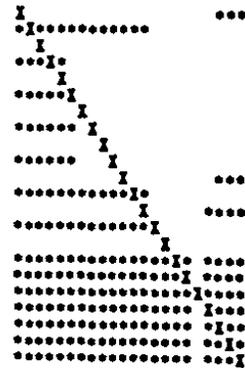


Figure 15: T_{ij} Yield Map

5 Conclusions

The ASSOCMEM represents a novel computational metaphor for VLSI. Both the implementation and the functional behaviour of this chip are radically different from conventional digital architectures and algorithms. By implementing a well understood system, such as associative memories, with collective circuit design, we have produced a functional component demonstrating the usefulness of these architectures for solving ill-defined problems. Collective circuits are ideally suited to VLSI, as they require very large numbers of individually simple elements.

In the future, we expect to apply this approach to other applications, including analysis of "fuzzy" sensory data. Development of alternate circuit techniques is also proceeding, with the recent fabrication of a CMOS associative memory based on slightly different principles.

Acknowledgements

This research was funded by the System Development Foundation. One of us (MS) was also partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

The authors are indebted to Dr. John Hopfield, David Feinstein, and John Platt for many hours of stimulating discussions concerning the nature and properties of collective systems.

References

1. Hayes, J.P., Computer Architecture and Organization, McGraw-Hill, 1978, Sec. 5.3.3.
2. Parhami, B., "Associative Memories and Processors: An Overview and Selected Bibliography", (1979) *Proc. IEEE*, Vol. 61, 722-723.
3. Weems, C. et al, "TITANIC: A VLSI Based Content Addressable Parallel Array Processor", (1982) *Proc. IEEE ICC*, 236-239.
4. Hopfield, J. J., "Collective Processing and Neural States," *Modelling and Analysis in Biomedicine*, 371-389.
5. Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons," (1982) *Proc. Natl. Acad. Sci. USA* 81, 3088-3092.
6. Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," (1982) *Proc. Natl. Acad. Sci. USA* 79, 2554-2558.
7. Feinstein, D. and Hopfield, J. J., private communication.