A digital CMOS fully connected neural network with in-circuit learning capability and automatic identification of spurious attractors.

Jean-Dominique Gascuel, Michel Weinfeld, Sami Chakroun
Ecole polytechnique
Laboratoire d'informatique
F-91128 Palaiseau Cedex
France
gascuel@polytechnique.fr

Abstract

We describe a completely connected feedback network with 64 binary neurons, using digital CMOS technology. The architecture implements a linear systolic loop, in which each neuron stores locally its own synaptic coefficients, and the potential calculation needs N time steps, each performing N partial weighted sums, to realize the N^2 operations needed. It implements internal learning capabilities, using the Widrow-Hoff rule, which converges towards the pseudo-inverse rule by iteration, thus allowing partial correlation between prototypes, and a higher capacity, compared to the Hebb rule. Also, it implements an internal mechanism for detecting relaxations on spurious states. The average retrieval speed is about 20 μs, whereas the learning time is approximately 15 to 30 ms for 15 moderately correlated prototypes.

1 Introduction

We have designed a fully digital 64 neurons Hopfield type network[1]. Several unique and original features, which are not found on other comparable designs [2,3,4], are imbedded in this circuit, making it a good candidate for higher level autoadaptive architectures:

- internal learning, implementing the Widrow-Hoff scalar and local rule, which iteratively converges towards the non-local pseudo-inverse rule, allowing storage of moderately correlated patterns,
- internal detection of convergence on spurious states, allowing automatic routing of the information processed by the network to other levels of a multi-network architecture,
- internal random perturbation of the network (roughly equivalent to some kind of annealing), which can be triggered by the spurious states detection, allowing a significant overall improvement of the attractivity of the network stored memories.

Our circuit does not require computations from a host, either in the learning or in the retrieving phase. This is consistent with our goals, which are to build autonomous hierarchies of elementary networks, endowed with some kind of low-level cognitive functions. In these architectures, massive parallelism and information processing capabilities can be implemented by many of these low level circuits, instead of trying to design huge flat hierarchy networks, with rather severe architectural and technological limitations.

2 Basic choices

We use full digital circuitry, in order to avoid the difficulties met by analog implementations, having insufficient precision and some physical shortcomings. The only real difficulties arise from the connectivity issues. To solve this problem, we use a linear systolic loop architecture [5,6], in which each neuron has its own synaptic coefficients (one column of the synaptic matrix) stored locally in a circular shift register (figure 1). This choice limits and simplifies the data lines connecting each neuron to the rest of the network : designing such a circuit consists mainly in designing the elemen-

Paper published at *IEEE Conference on Euro ASIC*, Paris, 1991.

tary neuron, which can be duplicated as a macrocell with automatic connection of the signal ports.

3 The detection of spurious states

In the context of auto-associative memory applications, we have designed the network with the ability to signal by itself whether it has or not succeeded in recognizing a pattern (since a network gives always an output for every input stimulus), producing a binary signal indicating whether the final state is a stored pattern or a spurious state. The vectors presented for learning by the network are divided into two fields: the longer contains the pattern itself, and the other one (six bits wide), called the label, contains the result of the coding of the first through a cyclic error correcting code. The whole vector (pattern concatenated to its label) is stored. In the retrieval phase, the attractor on which the network has relaxed is submitted to the same coding: if the information-carrying field is coherent with the label-carrying field, this is a strong indication (≈98% confidence) that this attractor corresponds to a stored pattern, and vice-versa.

4 Implementation

As the synaptic memory needs a rather low number of words (64 × 9 bits in each neuron memory, this minimal precision having been computed by simulations [7]), and since the synaptic coefficients are always stored or read in a serial fashion, we designed a custom 9 bit wide circular dynamic tristate shift register, having a power consumption and an area lower than that of conventional silicon compiler generated RAM. The rest of the neuron (control, arithmetic and logic unit) has been designed with standard cells.

Since the neuron states are only coded on one bit, the operating part is very simple: it uses fixed-point signed arithmetic, twelve-bit wide. The only operations implemented are addition, complementation and arithmetic shift. Provision is made to take into account addition over or underflow, which is transformed into saturations, so as to avoid errors.

For convergence detection, each neuron stores its previous state in a one bit local register, which is xored with the new state bit at the end of an updating cycle, in the relaxation phase. The local results in each neuron are just combinatorially ored, giving a low true

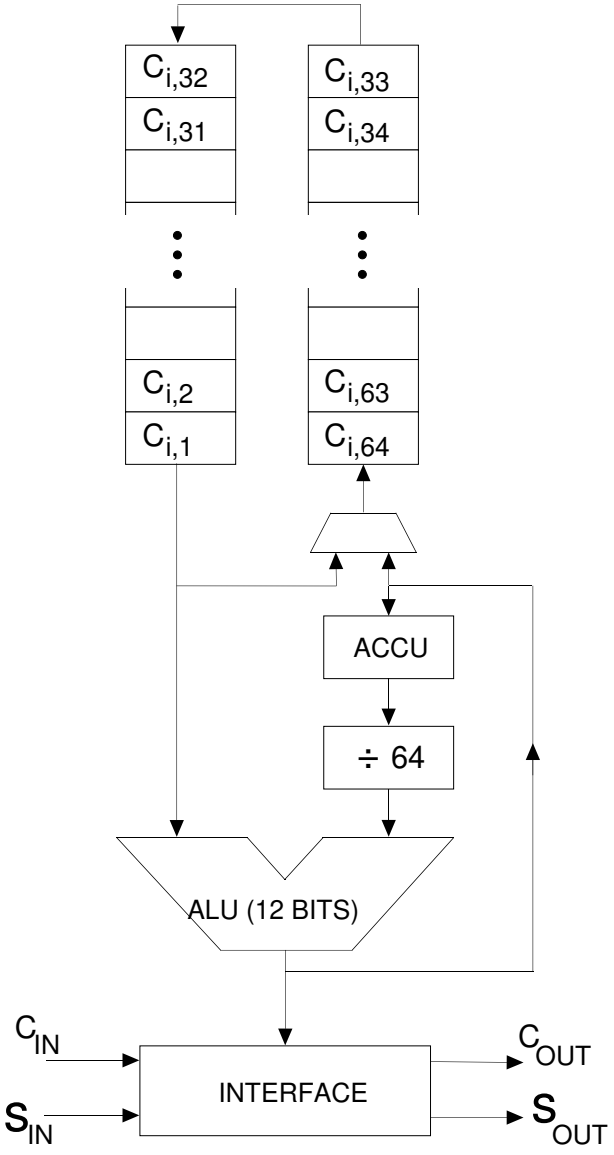

Figure 1: Simplified schematics of one neuron. S is the component of the network state vector (sequential), C is the convergence signal (combinatorial), the C_{ij} are the synaptic coefficients. Common command and clock signals have been omitted for clarity.

signal when the whole network is stable. Thus, an extra updating cycle is needed, where no updating actually takes place, just for asserting the convergence, but during which the serial state vector is converted into a parallel output. In the learning phase, since each neuron calculates a synaptic coefficient increment, the six most significant bits are ored to give a local convergence signal (null increment), which is in turn ored through all neurons, the same way as in the relaxation stage (it is not necessary to use the other less significant bits since a subsequent division trims them anyway).

The random generator needed by the internal annealing capability uses a combination of two linear feedback shift registers to generate a serial string of the desired number (1 or 2) of randomly located bits which are xored to the state vector as required. Input/output operations are performed in parallel, the serialization of the data taking place inside the chip. It is possible, for use with external 16 bits buses, to multiplex the 64 bits in four 16 bits blocks: this is the reason why the WRen (write enable) and the READen (read enable) are four-bit wide, each bit enabling the corresponding block while the three others are kept at high impedance (figure 2). In addition to these I/O control signals, there are several main external control signals, plus some internal state signals, mainly for debugging purposes.

The overall duration of one updating cycle (in the learning as in the retrieval phase) is the product of the time required to accumulate an increment of the neuron potential (internal neuron computing delay) plus the time needed to shift the state vector by one bit, by the total number of neurons. With a basic cycle of 80 ns, this gives an internal computing delay of 60 ns, a shift delay (including various housekeeping tasks) of 20 ns, giving a complete updating cycle in 5.4 μs. In the retrieval phase, if we suppose that typically four cycles are needed (this number depends somehow on the initial Hamming distance of the stimulus to the corresponding stored pattern), counting the extra cycle during which convergence is assessed, one pattern recognition is performed in less than 22 μs. In the learning phase, the speed is lower for two reasons. The first is that the calculation of the coefficients increments needs two cycles: one for the calculation of each neuron activation, and the second for the calculation of the increments themselves. The second reason is that one has to reach the convergence of the coefficients to stable values corresponding to the exact learning rule: the numbers of iterations needed is roughly proportional to the number of stored patterns. Thus, in the same conditions as above, learning 15 moderately correlated patterns may take approximately 3 ms, which is close to an absolute maximum, corresponding to a rather high learning ratio ($\approx$25%). Compared to simulations, this is approximately 1500 times faster than an MC68030 workstation, and 6 times faster than a Connection Machine^{TM}, with equivalent operating parameters.

The circuit totals 420,000 transistors for an overall surface of 1 cm^2, and has been manufactured using a 1.2 μm CMOS double metal technology (figure 3).

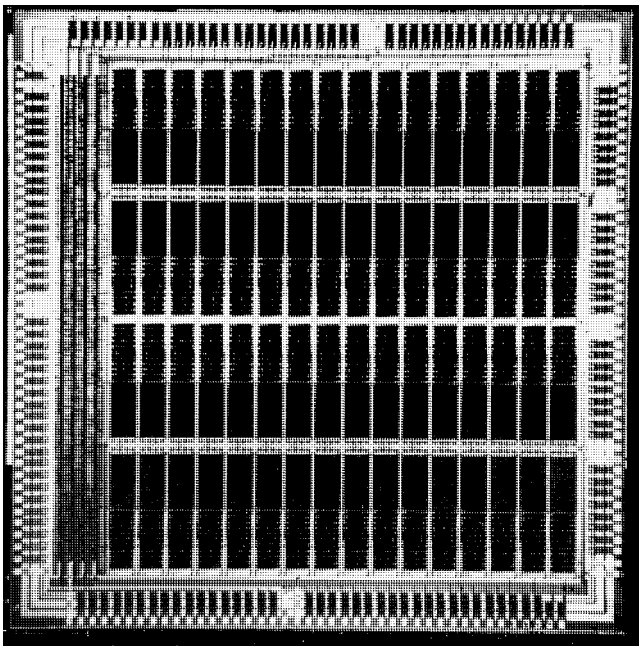

Figure 3: view of the chip (1x1 cm^2). The neurons are arranged in 4 rows of 16. The surface of each neuron is divided in two approximately equal parts : the memory (darker) and the ALU. The I/O and control circuitry can be seen to the left of the chip.

A first fabrication run has shown that the circuit was operational within specifications, at the nominal 20 MHz external clock. Due to an insufficient yield, a second run is currently in fabrication state, at this paper's writing.

5 A flexible simulator

The circuits are wired on Apple MCP^{TM} NuBus interface cards, to be configured into various multi-networks structures by software. Several such cards can be used,

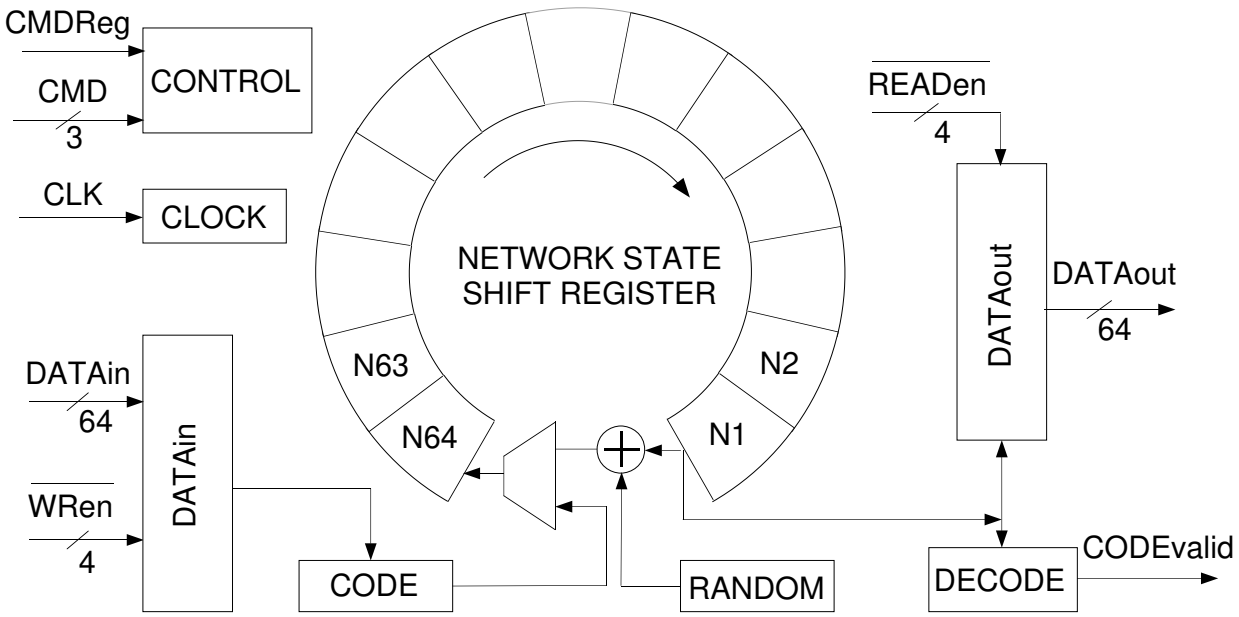

Figure 2: Schematics of the chip. Most signals have self-explanatory names. The CODE and DECODE blocks and the CODEvalid signal implement the spurious state detection, the RANDOM block implements the random bit string generation for annealing purposes.

needing not to sit in the same microcomputers, as long as these are linked by the standard LocalTalk™ network. This very convenient way of using many circuits for experimentation, under control of a simple human interface, should help assess various higher level architectures, which would otherwise be impossible to simulate just by software, except on very costly and very bulky superparallel mainframes.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[2] J. Alspector, R. B. Allen, V. Hu, and S. Satyanarayana, "Stochastic learning networks and their electronic implementation," in *Neural Information Processing Systems, Denver, Co 1987*, (D. Z. Anderson, ed.), (New York), pp. 9–21, American Institute of Physics, 1988.

[3] D. B. Schwartz, R. E. Howard, and W. E. Hubbard, "A programmable analog neural network chip," *IEEE Journal of Solid-state Circuits*, vol. 24, pp. 313–319, April 1989.

[4] J. Theeten, M. Duranton, N. Mauduit, and J. Sirat, "The LNEURO-CHIP : a digital VLSI with on-chip learning mechanism," in *International Neural Network Conference*, INNS, Paris, July 1990.

[5] S. Y. Kung and J. N. Hwang, "Parallel architectures for artificial neural nets," in *IEEE International conference on neural networks*, pp. 165–172, San Diego, 1988.

[6] M. Weinfeld, "A fully digital CMOS integrated Hopfield network including the learning algorithm," in *VLSI for Artificial Intelligence*, (J. G. Delgado and W. R. Moore, eds.), Kluwer Academic Publishers, 1989. International Workshop on VLSI for Artificial Intelligence, Oxford, July 1988.

[7] P. Alla, G. Dreyfus, J. Gascuel, A. Johannet, L. Personnaz, J. Roman, and M. Weinfeld, "Silicon integratiom of learning algorithm and other auto-adaptive properties in a digital feedback neural network," in *1st International Workshop on Microelectronics for Neural Networks*, IEEE/ITG, June 1990. Dortmund.