A Software Reconfigurable Multi-Networks Simulator Using a Custom Associative Chip

Jean-Dominique Gascuel, Eric Delaunay, Lionel Montoliu,
Bahram Moobed, Michel Weinfeld

Ecole Polytechnique
Laboratoire d'Informatique
91128 Palaiseau Cedex
France
gascuel@poly.polytechnique.fr

Abstract

We have developed a special purpose simulator, designed to try various interconnection schemes between several similar associative chips, to assess hierarchical assemblies of neural networks. These chips are digital feedback networks with 64 fully interconnected binary neurons, capable of on-chip learning and automatic detection of spurious attractors, that we have designed and fabricated. This simulator is based upon the MCP^{TM} development board supplied by Apple Computer, to be used in the Macintosh II family. Each such board can house four associative chips, which can be independently accessed from the microcomputer through a message-passing scheme. One machine can interface up to five such boards. The simulator is designed to transparently address chips not only inside the machine from where it is run, but also chips residing in other machines, through the LocalTalk^{TM} network. This means that the number of chips that can be used is high (of the order of several tens), and that several independent simulations can be run simultaneously, provided that they use different set of chips available on the local area network. All the virtual interconnections between chips are made at the neuron level, which means that the individual components of binary vectors processed by each chip can be routed to the input or from the output of any other chip. In addition, the simulator scheduling allows to provide sequentiality in the information processing.

1 Introduction

Artificial neural networks open wide areas of investigation leading to efficient information processing systems, which are expected to behave in a way somehow similar to their biological counterparts. Nevertheless, most efforts have been devoted to finding paradigms, such as multi-layer or dynamic networks, to name a few, and to strive for improvements in their internal algorithms and structures. These efforts have produced very interesting results, but still a fundamental property of neural networks remains, namely the unavoidable residual errors. These errors, resulting from several constraints, altogether theoretical and practical, make the use of a unique neural network for decision processes or pattern recognition safe only on the long run, that is to say when many experiments have been performed on real unknown sets of data, results statistically conforming to the training. Similarly, the complexity of what can be performed by any network is obviously limited : making the learning rules and internal structure more complicated to cope with more difficult tasks, lead to networks which are not only impossible to implement, but even to simulate without excessive difficulty.

Paper to be presented at the *Intl. Joint Conference on Neural Network*, Baltimore, 1992.

Biological and psychological evidence shows that the natural information processing is done in a very large amount of functional units, each one highly specialized in some particular simple task, these units working mostly in parallel, but also with mutual time delays or synchronization. Moreover, from the sensory inputs to the innermost representations in the brain, the information undergoes continual processing through these concurrent units and subsystems [1]. Another important property of this processing is that it is error prone, but it includes enough various correcting mechanisms so that it can assure a global acceptable behavior. On a short time scale, these mechanisms are amenable to ordinary feedback, whereas on a long time scale they possibly imply structural modifications. Both can be grouped under the important notion of auto-adaptivity.

Hence, it is only natural to think of artificial neural processing architectures in terms of many low-level networks, each having a rather high connectivity, more loosely interconnected to form some kind of oriented trellis [2]. These trellis, being adaptive, can modify themselves by local learning in the networks that are their basic constituents, and also in their local or global topology by modification of the paths between these networks. The networks themselves need not be all the same kind, need not be all adaptive, and some constituents can even be only simple non-connectionnist structures, such as for instance majority voters, ordinary memories or logical operators.

As it is well known that the simulation of the simplest networks is already a non trivial task (especially when taking learning into account), it seems difficult to expect simulating the above mentioned structures without a very high amount of efforts and computational resources. A better and promising solution consists in hardwiring the basic networks, letting to software only the care of interconnecting them. Later, whenever complex interesting architectures have been found, it will prove more profitable to hardwire them to obtain the maximum processing power and more compactness.

To put these ideas in practice, we have built a software reconfigurable simulator, based on a custom associative memory chip that we have previously designed. We shall describe successively the chip, then the simulator.

2 The auto-adaptive associative chip

2.1 General properties

The chip represents a fully digital 64 binary neurons Hopfield type network. Several unique and original features imbedded in this circuit make it a good candidate for higher level auto-adaptive architectures [2]:

- internal learning, implementing the Widrow-Hoff scalar local rule, which iteratively converges towards the non-local projection 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.

The architecture is a linear systolic loop [3, 4], in which each neuron has its own synaptic coefficients (one column of the synaptic matrix) stored locally in a circular shift register. 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. 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. It is encapsulated in a PGA176 package.

2.2 Automatic recognition of spurious attractors.

The network his able 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. This property is provided by using a small part of the vectors as a label, deduced from the rest of the vector by a coding through a cyclic redundancy code. This coherence, only found on attractors corresponding to learnt prototypes, provides a 98% confidence identification with a 6 bits code field. [5, 6, 7]

2.3 Speed of operation.

The overall duration of one updating cycle, with a basic internal clock cycle of 100 ns (20 MHz external clock), is 6.4 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 30 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%). This speed is compatible with most real-time stand alone applications, and with the timing of the simulator which will be described now.

3 The simulator

3.1 General features.

Since the higher level architectures to be implemented must be very diverse, we need a great deal of flexibility : the number of basic networks to be interconnected is variable, the interconnection must be completely configurable, which means that for instance each component (bit) of the input or the output vector of a network must be separately accessed and connected to any other component. So we have chosen a flexible environment : a chip is wired onto a standard intelligent interface board, this board inserted in a microcomputer. Several chips can "talk" to each other through the micro-computer bus, and if several microcomputers are linked through a local area network, chips on different machines can as well be "connected" together and also communicate with memory buffers and/or disk files. Using standard interfaces, software and operating systems, relieves us from all low-level implementation chores, allows minimal development efforts, and provides good opportunities for future upgrades (chip as well as host machine).

The simulator allows to implement sequentiality in the information processing, that is for instance making some chip wait relevant inputs before beginning a relaxation, and so on. It is also possible to operate several independent instances of the simulator, from different machines on the same network, provided there are enough chips available : each instance "books" the necessary chips among those seen as being free on the local area network.

The speed of the simulator depends on many factors : number and location of the chips, complexity of the virtual communication links between them. In no case is the intrinsic chip speed a limitation, but rather the message-passing overhead. Nevertheless, the expected goal is reached : the software load is shifted onto a higher level of complexity than the basic neural network, which has become an almost ordinary component.

3.2 The basic interfacing

It consists in a MCP^{TM} board (Macintosh Coprocessor Platform) supplied by Apple Computer Corp. This board includes a M68000 microprocessor, memory, and specific interface ASICs. Custom circuitry connected to the microprocessor bus on the board can be accessed by the host microcomputer through a board-resident real time system called A/ROSE^{TM} (Apple Real Time Operating System Environment), exchanging information using a message-passing scheme. There is enough space on the board to accommodate four independent associative chips, occupying different addresses in the memory space of the M68000. Using a Macintosh II family computer allows to interface five such boards in one machine, that is 20 chips. Several machines linked by the LocalTalk^{TM} proprietary network can thus allow the use of many more chips.

3.3 The software.

The simulator software is organized into several distinct layers, each of them having a dependency only with its predecessor.

3.3.1 Board level.

The board-resident software has the following main features :

- time sharing management of concurrent tasks residing on the boards processor ;
- passing of messages between different tasks, whether these tasks are local to the board or run in another board in the same machine.

The reentrant main program interfaces each of the four chips with a message queue, which include data and commands :

- test and initialization of the chip ;
- reception of commands to be executed together with the relevant data ;
- micro-code generation ;
- output of results.

Since the switching time of a task is approximately 135 μs, it would be time-consuming to switch constantly from one task to another, each corresponding to different chips. Thus, the chip managing task works as a blocked task, in the learning as well as in the relaxation phase. Informations is grouped into packets (580 bytes long, as determined by the scheduling machine resident task, see later), so the processor is allowed to switch between tasks only during the interval between two packets. This strategy allows some kind of parallelism between the control of a specific chip, and the packets transmission between chips on the same board, or between boards.

Since the routing of the data must be done at the bit level, the 64 bit vector present at the output of a chip, together with two control bits (spurious state detection bit and identity bit) are cut into nine bytes, each treated as independent, and encapsulated into nine different packets. These packets are broadcast to all other chips that are needing even only one bit of a particular byte. At the input of a circuit, the task has to reconstruct a 64 bits vector from information gathered into many different packets ; at this moment, it is possible to perform some logical operations between the received bits, if necessary. This overall strategy ensures a good compromise between the granularity of the transmissions (the coarser, the faster) and the granularity of the data, at bit level.

3.3.2 Machine level : the forwarder and the transport layer

The A/ROSE environment supplies a background task in the host machine. This particular task interfaces the messages issued by the boards to the LocalTalk^{TM} local area network, allowing the extension of the simulator between different machines. Each machine hosts such a task, all the remote ones being clients of the local one.

3.3.3 The man-machine interface

We use the basic system of the Macintosh family to implement a graphic interface, providing tools to configure the required structure, without too much concern for the physical location of the needed chips (except when overall speed is to be taken into account, see below). When the connectivity of the simulator has been set, the data needed for learning, or the stimuli to be processed can also be entered through this interface. To avoid too much overhead, it is also possible to declare the structure and the data using a simple language, and store or retrieve data and results in ordinary disk text files. This last feature can be used as a mean of communication between the whole simulator and any other Unix machine, through Ethernet networking.

3.4 Performances

We have tested a network made of four chips, each of them being in a different machine, with a connectivity represented by eight links of one byte wide. One packet takes approximately 500 ms to travel, this time being imposed by the LocalTalk^{TM} network (as said earlier, the intrinsic chip speed is not at all a limiting factor). So clearly the user must use some care when configuring his architecture, to confine the heavily linked chips in the same machine. Nevertheless, the simplicity of the simulator makes it a much more comfortable and flexible solution than any full software simulation of similar multi-networks. For the future, it is probable that fully wired boards using integrated chips (before the advent of fully integrated systems), will be the solution for optimal speed and compactness, but still the use of an efficient simulator, such as the one presented here, should be necessary as a first step.

4 Some examples of high level architectures

A rather simple example of multi-network association is a so-called "multi-agent" classifier. In most connectionnist classification schemes, a single multi-layer network is used, with the well-known back-propagation learning. Its use implies a complete sequence of presentation of many patterns belonging to each class to be later separated. The structure of the network results from a rather tedious procedure, where the size and number of layers must be found somehow by trial and error to satisfy the learning and generalization criteria. Later, the inclusion of new knowledge may require a complete re-learning sequence, and even perhaps a modification of the the structure of the network.

We suggest using several associative networks, each one of them learning various examples of the same class. In the recognition phase, an unknown input is fed in parallel to all of them, the answer comes from the network that recognizes a known pattern, whereas all or a majority of the other networks relax on a spurious attractor. In no case the particular pattern on which one network has relaxed, is of interest. The answer of the whole system is a bit pattern composed of the known pattern/spurious attractor bit of each network, this bit pattern containing in general only one bit raised to 1, all the others being at 0. The ambiguity, where more than one bit is on, is a significant case that must be normally taken into account by the rest of the system. Among the advantages of this kind of classifier, we have just seen the simplicity of ambiguity detection. Another important advantage is that a new class can be learnt by just adding a new network and having it learn this particular class, without disturbing any of the other networks. This procedure can even be made partly or totally automatic.

Another example of elementary network association we are currently working on (intended for image recognition), consists in a quad-tree structure, each node being an identical classifier made of several associative network, as described above. Each level in the tree corresponds to a different resolution of the picture, the recognition beginning with the root network, at the lowest resolution. All clusters store identical elementary features, as for instance left-right or top-down contrasts, vertical, horizontal and oblique lines, etc., and receive all together the fraction of the picture corresponding to their level in the tree. The response of the whole system is a vector, the components of which (the identification bits of each network) characterize the presence of an elementary feature at a certain level of resolution. This hierarchical description of a picture is of course much more efficient and compact than any "flat" representation of equivalent or even longer number of bits. In addition, it is obtained from the independent and simultaneous relaxation of many associative networks, so it is fast.

5 Conclusion

We have shown briefly that it is possible to interconnect elementary networks to obtain more complex architectures, able to deal with information processing problems that isolated networks have difficulties to cope with. Provided that these elementary networks have the necessary properties for implementing auto-adaptive systems, and that they do not induce computational overhead by themselves, a software reconfigurable simulator is capable to address a higher level of complexity, probably closer to the complexity of some cognitive systems (even if there is certainly many more levels to get over). To assure this goal, integrated custom neural chips are obviously one good solution, since the advances of technology make them easier to produce and to use.

References

[1] W. J. Freeman. A physiological hypothesis of perception. *Perspectives in biology and medecine*, 24:561–592, 1981.

[2] M. Weinfeld. Integrated artificial neural networks : components for higher level architectures with new properties. In F. Fogelman and J. Hérault, editors, *NATO advanced workshop on neurocomputing*. Springer Verlag, 1990.

[3] S. Y. Kung and J. N. Hwang. Parallel architectures for artificial neural nets. In *IEEE International conference on neural networks*, volume II, pages 165–172. San Diego, 1988.

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

[5] J. D. Gascuel and M. Weinfeld. A digital cmos fully connected neural network with in-circuit learning capability and automatic identification of spurious attarctors. In *International Joint Conference on Neural Networks*, July 1991.

[6] P.Y. Alla, G. Dreyfus, J.D. 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.

[7] A. Johannet, L. Personnaz, G. Dreyfus, J.-D. Gascuel, and M. Weinfeld. Specification and implementation of a digital hopfield-type associative memory, with on-chip trainning. *IEEE Trans. on Neural Networks*, to appear 1992.