

NEURAL NETWORKS ON SILICON: THE MAPPING OF HARDWARE FAULTS ONTO BEHAVIORAL ERRORS

Vincenzo Piuri, Mariagiovanna Sami, Renato Stefanelli
Dipartimento di Elettronica, Politecnico di Milano
Piazza L. da Vinci 32, I-20133 Milano, Italy

Abstract

The problem of defect- and fault-tolerance in neural networks becomes increasingly important as a growing number of silicon implementations become available and mission-critical applications are envisioned. As an alternative to architecture-specific policies, intrinsic characteristics of the neural paradigm with respect to a functional error model are considered. In particular, this has been done for multi-layered back-propagation networks, where both the classification errors induced by insurgence of a fault and the possibility of masking such errors through a repeated learning phase have been studied.

Such abstract results can be used to analyze various silicon architectures implementing the multi-layered nets; physical faults are mapped onto the functional error classes, so as to evaluate both the intrinsic robustness of the various architectures and their critical areas, where ad-hoc design modifications or redundancies must be inserted to increase fault-tolerance properties. In the present paper some relevant implementations, representative of various design philosophies, are considered from this point of view.

Introduction

A number of silicon implementations of neural networks have been presented by now in the literature, and various devices are even commercially available. Alternative approaches have been proposed, differing not only in the implementation technology adopted (analog vs. digital solutions) but also with respect to the philosophy underlying the realization. Thus, while some architectures simulate net's functions (so that no one-to-one correspondence between single neuron's functions and a corresponding component functions can be created) others emulate the individual neurons; again, while in some solutions the complex neural connectivity is mapped onto an identical physical connectivity [1] in other solutions the logical data transfers are time-multiplexed onto a simpler physical interconnection structure (see e.g. [2 — 9]).

Since ever larger devices are considered (in particular, WSI solutions have been advocated [8], [9]) and possibly mission-critical applications are foreseen, it becomes mandatory to take into account aspects such as tolerance to production defects and/or to run-time faults. Some proposals on this subject have been presented in current literature: two different points of view are usually considered. In some instances the aspects taken into account were purely those pertaining to the *implementation architecture*; this approach is typical of digital implementations, and actually it does not exploit the characteristics of the neural paradigm but optimizes redundancy allocation and reconfiguration strategies with respect to the distributed architecture adopted. Examples of such philosophy are given in [10] and in [11]. The neural computation is taken into account only insofar as its mapping onto an array architecture is affected by the presence of faults — or, alternatively, such mapping can be such as to afford a simple solution to the reconfiguration problem.

On the other hand, it is possible to consider specifically the characteristics of a neural net *per se*, taking into account therefore the behavioral errors rather than the logical faults or the physical defects of a specific implementation. In that case, abstract computational errors located in a neuron or in a synapsis are modelled, and the effect of various distributions of such errors on the network's behavior are evaluated. This second approach was undertaken, for example, in [12] and in [13]; in this second paper, the correspondence between behavioral errors and physical faults affecting an analog implementation was also examined. Both papers discuss the effect of errors and do not propose strategies for overcoming them, apart from the (possible) intrinsic tolerance of the neural net itself.

In [12], the instance of multi-layered back-propagation network [14] was in particular examined; the error model introduced allows for different errors in the behavior of neurons (such as incapacity of correctly performing the summation of weighted inputs or of correctly evaluating the non-linear function) and of the synapses (errors affecting the values of weights or of the weight-signal product). Since no reference is made to specific implementation technologies, general types of errors must be accounted for — e.g., both percentage variations and stuck-at errors on weight values. The effect of such errors on the network's classification capacity was examined both from a theoretical point of view and through extensive simulations. Both theoretical and simulation results allowed to state that occurrence of an error in the network can degrade the net's classification capacity even in a very relevant way; as might be expected, the effect is more relevant for networks near the "minimum configuration" for a given set of training patterns (i.e., the smallest networks capable of correctly performing a given classification), since no intrinsic redundancy is present, and more relevant when it affects neurons in the output layer. Further comparisons were made for networks of different topologies — i.e., with different numbers of layers, with comparable numbers of neurons but different distributions among the layers, etc. An example of such simulation results is given in figure 1.a, where the effects of an error in a single neuron or synapsis for a three-layer net with increasing numbers of neurons in the

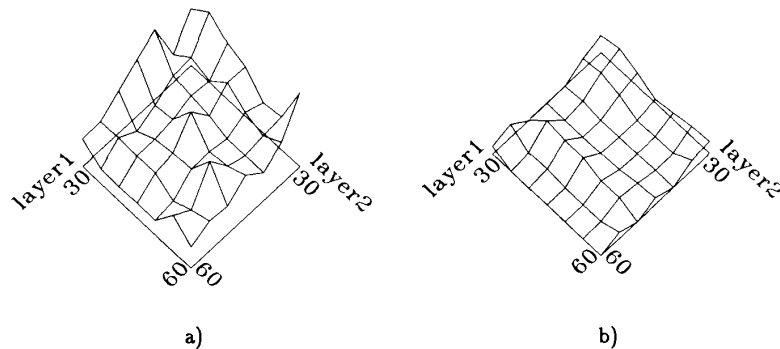


Figure 1 - Evaluation of the effects of errors onto the neural computation: after error occurrence (a) and after repeated learning (b).

input and hidden layers are shown (all nets are geared to classification over the same set of input pattern classes).

In [15] it has been proved that — for networks provided with sufficient intrinsic redundancy — effects of an error can be much lower if the learning phase has been continued beyond perfection point, so as to bring the operating point of each neuron well into the saturation region of the non-linear function. This in fact superimposes a sort of "information redundancy" over the intrinsic structure redundancy of the network. (It should be noted that completely connected networks are consistently taken into account, so that a redundant number of neurons involves also redundancy in the set of synapses and correspondingly of synaptic weights). Mathematical conditions allowing the net to overcome the effects of the error through a suitable redistribution of weights in the error-free neurons were introduced in [12]; such weight re-distribution constitutes in fact a "repeated learning" phase. While analytical computation of the modified weights would be extremely cumbersome, it can be naturally provided by an implementation capable of supporting supervised learning: this repeated learning phase can be considered as a particular instance of "time redundancy", introduced only in a sort of reconfiguration phase and not affecting the subsequent recall activities. A quantitative evaluation of the effects of repeated learning was obtained through simulations: figure 1.b shows the residue classification error, after a re-learning pass has been completed, for the same nets examined in figure 1.a. As it was to be expected, re-learning allows to reduce the effects of the error increasingly with increasing numbers of neurons in the net, i.e., with increasing intrinsic redundancy.

In the present paper, we start from the purely behavioral analysis described above and examine some specific implementations of neural nets described in current

literature from the point of view of the behavioral errors. To this end, we examine how hardware faults map onto the behavioral errors, and thus derive the impact of hardware faults onto the computational capacity of the whole system. The behavioral errors we take into account affect, respectively:

- a. errors affecting the values of the synaptic weights w_{ij} ;
- b. errors affecting products $w_{ij}x_j$;
- c. errors affecting the value of the input summation of an individual neuron;
- d. errors affecting the final evaluation of the non-linear function and, as a consequence, the output signal provided by the neuron.

We assume initial learning (prior to fault occurrence) to have been perfected until the null-error condition has been reached; therefore, hardware faults may be expected to result in an immediate degradation of the system's behavior. Analysis of the fault-to-error mapping allows us to state whether the implementation has the same robustness characteristics as the abstract network model (and, therefore, the error can be masked through repeated learning) or else the degradation induced by the fault is too relevant to be overcome even by repeated learning. In this second case, we will identify the most critical points of the architecture, where architecture-specific redundancy policies must be adopted to increase the system's robustness.

Of the many silicon architectures, we concentrate on four, as representative of some main approach. In particular, we take into account the basic characteristics shared by most analog implementations (section 2), and three digital architectures, each characterized as an example for the solution of some typical implementation problems.

Implementation analysis: analog solutions

Many analog solutions have been proposed for implementation of various classes of neural networks; most share some basic characteristics relevant for our analysis of fault-to-error mapping, and in this section we will concentrate on them.

Analog architectures in general do not involve any form of time-multiplexing over the various components that implement the abstract operators present in the neural network. Thus, referring to the particular case of multi-layered nets:

- a. each neuron in layer i feeds its output signal on a dedicated line connecting to the neurons of layer $i + 1$;
- b. the signal-to-weight product is performed by a dedicated multiplying device associated with each "synapsis"; thus, for each neuron there are as many multipliers as incoming synapses;
- c. usually, the multiplying device is a simple MOS transistor (considered as a variable-resistance device) whose resistance value is controlled by a capacitor connected to its gate;
- d. the input summation to the non-linear function is achieved by a simple current node.

It is then evident that the behavioral errors listed in the previous section map directly, one-to-one, onto the physical faults that affect such a structure; in particular, failures of individual synapses or individual multipliers can well be modeled, and mutual independence between faulty devices (if multiple errors are to be taken into account) can safely be assumed. If the network is initially provided with a measure of intrinsic redundancy, fault of a physical component will not result in global failure but it will simply degrade the system's operation, leaving intact the possibility of recovery.

The above considerations hold, in fact, for the components that implement the "logical behavior" of the network; other electrical components must actually be considered, namely:

- * power supply
- * weight-value refresh circuit
- * in some instances, shifters used to extract the system's output values (this is the case, e.g., in Mead's retina [3]).

Power supply and output shifters are *global* system elements, and as such their failure would affect in a fatal way operation of the whole system; it may be observed anyway that such global system elements are found in any type of implementation (power supply in any case, system clock for digital synchronous solutions, etc.) and that for them dedicated, robust design techniques must be adopted (in particular, the case of robust power supply and clock design have been taken into account for WSI devices). As for weight refresh circuitry, the individual designs must be examined; while a section common to the whole architecture is always present, and therefore a fault located in such section would always lead to fatal failure, the refresh circuitry associated with each single weight would actually result in error of that weight only.

The above considerations can be applied also to the particular (presently quite few) instances of digital architectures where the intrinsic network parallelism is mapped onto the architecture without recurring to time multiplexing; such is the case of the solution presented in [1], where large pipelined binary trees are used to create the cells.

Implementation analysis: digital architectures

From a fault-tolerance point of view, the case of digital architectures has at the same time advantages and disadvantages by comparison with the analog solutions. While on the one hand problems such as reconfiguration have been extensively studied and satisfactory solutions can be found for various architectural instances, on the other hand digital implementations of neural nets in general make use of some form of time multiplexing so that single physical faults map onto a multiple behavioral errors, thus affecting even in relevant way the system's operation.

The first digital architecture here examined is the one presented by S.Y. Kung in [4] (see figure 2). This systolic-array structure can support the mapping of several

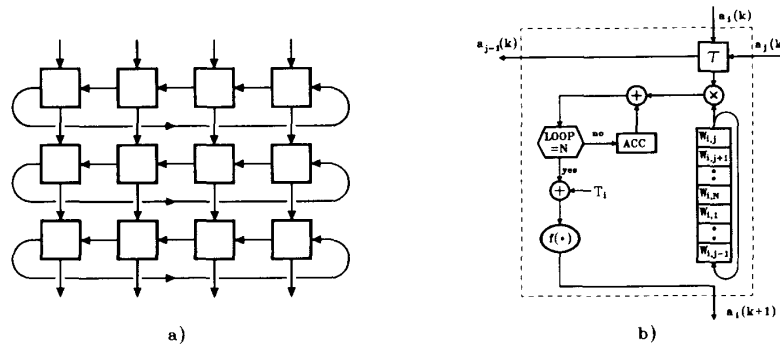


Figure 2 - Kung's digital architecture for neural networks

neural models; again, we consider in particular the case of multi-layered back-propagation networks. The mesh-connected array architecture is given in figure 2.a; one neuron is mapped onto one processing element, and a full layer is mapped onto each row. All neurons perform the evaluation of the non-linear function simultaneously, and thus they generate simultaneously their output signals. Each PE of a layer K feeds its output signal to the PE in the same column in layer $K+1$; inside each layer, the memory elements accepting the inputs from the preceding layer are then connected to create an equivalent recirculating shift register.

The simplified internal structure of a PE is given in figure 2.b. Suitable devices implement summation and evaluation of the non-linear function. The synaptic weights associated with the incoming synapses are also stored within the node, in a re-circulating memory whose circulation signal is synchronized with the activation signals propagated along the interconnection grid. (This re-circulating memory approach has been adopted by other authors as well, e.g. in [17]). Each node in a layer $K+1$ associates the incoming signal from layer K with the correct synaptic weight and then propagates the same signal, unchanged, to its horizontal neighbor along the direction of propagation. In the same way, a node receiving an "horizontal" input associates it with the correct synaptic weight, etc.: only when the whole memory has re-circulated will the non-linear function be finally evaluated.

This approach, that provides a compromise between the intrinsic parallelism of operation of a neural network and the serialization required by complexity requirements of the digital architecture (the number of multipliers and of inputs to the summation unit are kept at a minimum) requires first of all a correct ordering of weights in each synaptic memory, as well as a strict system synchronization. Given the internal structure of the neuron, the following considerations may be immediately made as far as the error model is concerned:

- a. a fault in even one word of the weight memory will affect all synaptic weights, given the re-circulating structure adopted;
- b. a multiplier fault will affect all products signal-weight.

As a consequence, all internal node faults affecting the neural *computation* result in a global neuron fault, and a strongly simplified model can be adopted for the simulations (referring simply to percentage or stuck-at faults on the neuron's outputs).

The problem is quite different if the internal fault refers to a node's capacity of forwarding to other nodes the activation signals received; in that case, a single fault degenerates in a global error. If the node is built in such a way as to "isolate" it whenever a fault is detected, still granting correct operation of the bypass switches that propagate the firing signals, node's faults can be simply mapped onto a stuck-at zero single-neuron error of the behavioral error model.

Obviously, the interconnection structure is now a critical point: two different instances need be identified. On the one hand, some links simply propagate the output signal of *one* node to a corresponding node in the subsequent layer (such are the vertical links in the example of figure 2). A fault on such a link, if suitable fail-safe design techniques have been adopted, will again appear as a stuck-at on the output of the origin node and as a consequence it will be collapsed into the single-neuron error class. (The design must be such as to grant that in the receiving node a null input signal will be multiplied by with the synaptic weight associated with the faulty neuron). Totally different, even catastrophic, consequences are on the contrary to be expected if a fault affects a link that propagates input signals throughout a whole layer: such is the case with the horizontal links in figure 2.a. Then, if one link fails *all* nodes in the layer will ultimately receive a whole sequence of incorrect activation signals, and a massive failure will ultimately result. This is due to the fact that time-multiplexing is adopted over the horizontal interconnection links, over which all activation signals fired from the previous layer are sequentially circulated. Considering the global circuit design, it can be noticed anyhow that the area required by such critical interconnection links is certainly much lower than that required by the processing elements and the weight memories constituting the nodes. This can support the hypothesis of "faultless interconnections" widely assumed when dealing with systolic array fault-tolerance, where usually faults are located in nodes; on the other hand, even when such assumption is not accepted the use of robust design or possibly of redundancy for the interconnection links will not increase in an unacceptable way the cost of the system.

The systolic array philosophy has been adopted by a large number of authors; in particular, switched-bus oriented architectures have been widely suggested. While it is quite obvious that in all such instances the buses become the hard-core of the architecture as far as reliability is concerned, fault-tolerance policies devised for this class of array may be adopted (although it might be recalled that faults arising in the interconnection network are usually excluded from reconfiguration policies, as much less probable than faults in processing nodes [16]). This philosophy was

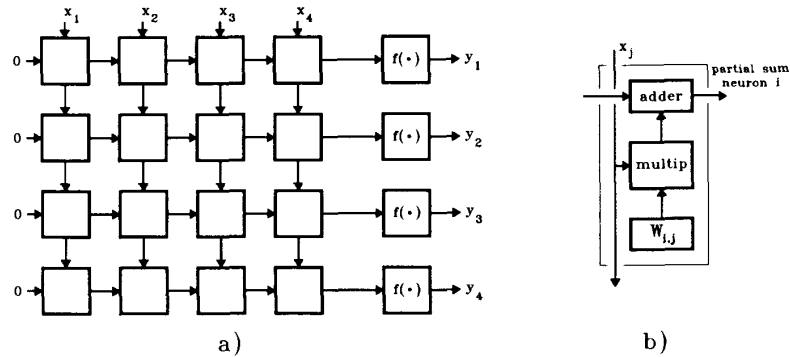


Figure 3 - The digital network by Lehmann and Blayo

adopted. e.g., in [11], where a first mapping of the neural network onto an ideal "strip architecture" (subsequently folded along the dimensions of a rectangular array) allows to overcome a comprehensive set of physical faults by exploiting the characteristics of layout and mapping at the same time. In that case, as in other similar ones, mapping of physical onto behavioral faults is not relevant, since the effects of faults are overcome at the architectural level independently of the semantics associated with the faulty devices.

A systolic array is also the basis of the solution proposed by Blayo and Lehmann [5], but the approach taken by these authors is quite different; here, in fact, simulation of a single neuron's computation by a subset of a systolic array is the chosen approach, so that a neuron's computation is distributed over a number of processing elements, rather than associating individual neurons with corresponding nodes of the array. A simplified structure of the architectures presented in [5], suited to implementation of feed-forward multi-layered nets, is given in figure 3.a. Here, the sum of weight-signals products for a given neuron is computed along an associated row of the array, while the rightmost column computes the evaluation functions. (The internal structure of a PE is given in figure 3.b). One weight is stored in a register inside each PE, and all PEs receive simultaneously the same input signals propagated along the column. The adder inside each PE updates the partial sum fed from the left-hand PE and then propagates it to the right-hand PE.

As a consequence, in particular, there is no mapping (possibly through time multiplexing) of synapses onto interconnection links. (In [5] the solution is analyzed in detail for implementation of generalized Hopfield and Kohonen models, but — as the authors themselves state — it is immediately extended to other networks, comprising the multi-layered back-propagation class, and to support such mappings a slightly more complex interconnection structure is provided).

While a single fault in the output column maps onto the error of a full neuron (the output signal is not correctly evaluated), a fault in any cell of the rectangular ar-

ray corresponds to an error in the summation input to evaluation of the activation function. Thus, the behavioral error model presented in the previous section has to be modified, accounting for percentage errors in the summation inputs or for errors in the non-linear evaluation function. It can be noticed that the relevance of such error depends not only on the proportional relevance of the value computed inside the node with respect to the total summation, but also (and this is more critical with respect to fault-tolerance) with the position of the node in the row. Still, it can be added that multiple faults within a row map onto a single behavioral error; moreover, even a fault in an interconnection link maps simply onto the error of one neuron. A more detailed fault analysis can be performed, both at node level (failure of the weight memory as distinct from failure of the arithmetic unit) and with respect to the behavioral errors these faults produce; if suitable fault-confinement policies are adopted for the circuit implementations, most faults can be reduced to "pruning" of a number of synapses leading to a specific neuron in the net. While such simple mapping of physical faults onto behavioral errors already grants an intrinsic robustness of this implementation approach, it can be further noticed that pruning of the complete multi-layered networks has been considered by several authors as a viable approach *from the conception* of the network implementation, allowing reduction of the weight memory dimensions (a main factor of silicon requirements when digital architectures are envisioned). Thus, presence of *production-time defects* that lead to such pruning can be overcome by a suitable initial distribution of weights within the network while still allowing acceptable system operation.

Concluding, the systolic array solution suggested in [5] appears to be far more intrinsically robust than the previous ones; in particular, there is no critical component whose failure is catastrophic. This can be seen as a consequence of two main factors: a single neuron's computation is distributed among several components, and no global communication structure multiplexed between a number of communicating elements is present.

As a final digital implementation, we consider the "neurocell array" structure presented in [6] (see figure 4). The structure there presented aims at a particular application environment, namely, that of dedicated, application-specific neural systems for which learning is perfected before field operation and for which silicon costs and processing speed are main factors of merit. Thus, while run-time (supervised) learning is not supported, the design optimizes such factors as the total memory requirements for synaptic weights (at system level) and the throughput during the recall phase for a continuous stream of input patterns, as well as the initial design costs; in fact, the neurocell array is proposed as a "semi-custom" device onto which the given application is mapped at production time by customizing interconnection routing, contents of the weight memories, control signals within the individual nodes. The analysis here performed can therefore be applied to this architecture to satisfy three possible goals:

- * verify the extent of classification errors as a consequence of run-time faults;
- * verify the residue yield of a device in the presence of end-of production defects;

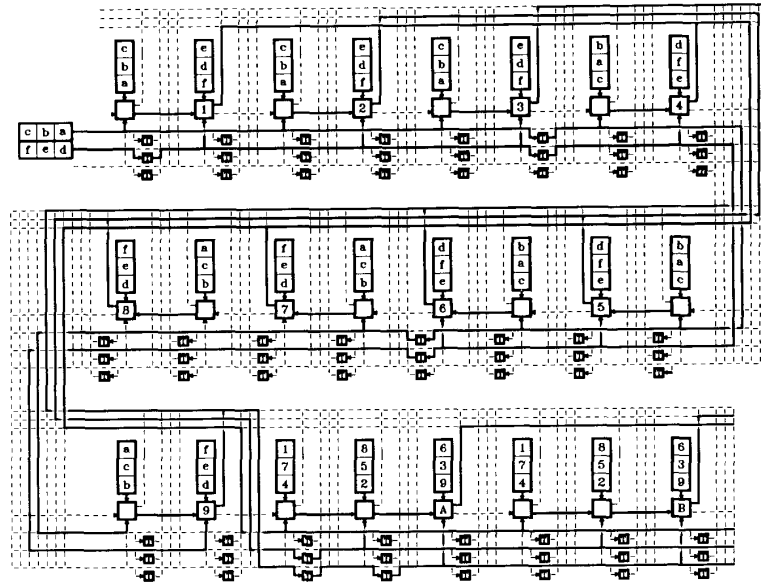


Figure 4 - The digital neural network by Distanti, Sami, Stefanelli and Storti-Gajani

in this case, modified weights will be evaluated off-line and then directly stored into the weight memories;

- * verify the possibility of a network's survival to run-time faults through weight update when a "field-programmable" technology has been adopted.

Let us briefly recall the basic philosophy underlying the neurocell-array operation. The array consists of "pseudo-neurons" provided each with a local weight (read-only) memory and with a processing unit which is either capable of evaluating the summation of weighted inputs (accumulated, if necessary, to a similar summation provided by a previous pseudo-neuron) and forwarding this value to the subsequent pseudo-neuron, or else of performing besides this "linear" operation also the evaluation of the non-linear function. Thus, the example in figure 4 shows a network with two layers, the first one consisting of nine neurons receiving six input signals, the second one consisting of two neurons only. Each neuro-cell can perform as a complete neuron, or as a pseudo-neuron in any position of a chain making up a complete (larger) neuron. Input signals to the various pseudo-neurons are fed through a stack of buses, through suitable delays, and output signals are fed by the final cell of each neuron onto one of a stack of output buses. Mapping of a multi-layered, fully-connected network involves a preliminary (and simple)

evaluation of the number of cells required to implement the neurons in each layer (evaluation is local to the layer), so as to optimize the total weight memory requirements as well, while pipelining of operation and of access to the various buses allows to optimize throughput. (Details of system operation and synchronization are given in [6]).

Let us now examine the physical faults that can characterize this array and their mapping onto behavioral errors. This can be summarized as follows:

1. a fault in one word of a weight memory maps onto a percentage error of a single synaptic weight; its effects (in general limited) can be simulated and — if the defect is present at production time — a modified weight distribution can be a priori evaluated;
2. a fault of the processing unit of a neurocell or a global fault of its weight memory (e.g., an addressing fault or an output-port fault) corresponds to an error in the summation input as far as the set of synaptic weights stored in that neurocell are concerned. If the design is such as to grant isolation of the faulty neurocell from the array buses (e.g., by cutting suitable links) then the result obtained is that of a "pruned" neural net in which one neuron of a given layer is connected only to a subset of the neurons in the previous layer. Such instance has been theoretically studied, as already said, and the presence of such a neuron can be taken into account while still obtaining a workable device. If the affected neurocell maps a whole neuron, this maps onto a single neuron's error and — by suitable design techniques — the error can be reduced to a well-defined one such as a stuck-at zero on its output;
3. a fault on an input bus corresponds to pruning of *all synapses coming from the same subset of neurons in the previous layer* leading to the subset of neurons fed by the bus itself. While a relevant error, it does not have the global effects seen previously in the instance of the systolic-array mapping; in fact, while time-multiplexing is present here also, it involves for each multiplexed bus only a subset of both origin and destination nodes. Unless both origin and destination nodes consist of single neurocells, a measure of computational capacity will survive and (depending on the intrinsic redundancy of the network) an attempt at modified weight evaluation can be made;
4. fault of an output bus corresponds to the failure of all neurons that feed their output signals onto this bus; As in the previous case, this is again a multiple but not necessarily a fatal failure.

Thus, intrinsic redundancy of the neural network can be exploited to achieve defect-tolerance of this solution. Further robustness can be achieved by a limited measure of structure redundancy, which can be very simply added given the basic linear-array structure of the neurocell array.

Concluding remarks

Analyzing silicon architectures to identify the mapping of physical faults onto behavioral errors allows first of all to verify whether the given architecture is characterized by the same robustness associated with the abstract neural model or whether it must be modified by introduction of suitable redundancies. Intrinsic redundancy of the neural model, or possibly information redundancy achieved during a prolonged learning phase, maintains its validity only if the physical faults of the implementation map one-to-one onto the behavioral errors of the model; this point is particularly critical for digital architectures, that in many case use some form of time multiplexing to solve the difficulties created by the very high connectivity requirements of the neural paradigm.

References

- [1] D. Zhang, G.A. Jullien, W.C. Miller, E. Swartzlander, "Arithmetic for Digital Neural Networks", *Proc. 10th IEEE Symp. Computer Arithmetics*, Grenoble, France, 1991
- [2] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proc. of the National Academy of Sciences, USA*, vol. 79, 1982
- [3] C. Mead, *Analog VLSI and neural systems*, Addison-Wesley Publishing Co., 1989
- [4] S.Y. Kung, "Parallel architectures for artificial neural nets", *Proc. Systolic Arrays 1988*, San Diego, CA, 1988
- [5] C. Lehmann, F. Blayo, "A VLSI implementation of a generic systolic synaptic building block for neural networks", *Proc. Int'l Workshop on VLSI for artificial intelligence and neural networks*, Oxford, UK, 1990
- [6] F. Distanto, M. Sami, R. Stefanelli, G. Storti-Gajani, "A compact and fast silicon implementation for layered neural nets", *Proc. IFIP Workshop on Silicon Architectures for Neural Networks*, St. Paul de Vence, France, 1990
- [7] J. Beichter, U. Ramacher, "VLSI design of a neural signal processor", *Proc. IFIP Workshop on Silicon Architectures for Neural Networks*, St. Paul de Vence, France, 1990
- [8] P.Y. Alla, G. Saucier, "Silicon integration of learning algorithms and other autoadaptive properties in a digital feedback neural network", *Proc. IFIP Workshop on Silicon Architectures for Neural Nets*, St. Paul de Vence, France, 1990
- [9] M. Yasunaga et al., "Design, fabrication and evaluation of a 5-inch wafer scale neural network LSI composed of 576 digital neurons", *Proc. IJCNN 1990*, San Diego, CA, 1990
- [10] L.C. Chu, B.W. Wah, "Fault tolerant neural networks with hybrid redundancy", *Proc. IJCNN 1990*, San Diego, CA, 1990

- [11] F. Distante, M. Sami, R. Stefanelli, G. Storti-Gajani, "Fault tolerance in massively parallel silicon architectures: the case of neural nets", *IEEE Proceedings*, in publication
- [12] V. Piuri, M. Sami, R. Stefanelli, "Fault tolerance in neural networks: theoretical analysis and simulation results", *Proc. Compeuro 1991*, Bologna, Italy, 1991
- [13] D.B.I. Feltham, W. Maly, "Behavioral modeling of physical defects in VLSI neural networks", *Proc. Int'l Workshop on Defect and Fault Tolerance in VLSI Systems*, Grenoble, France, 1990
- [14] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning internal representations by error propagation", *Parallel and distributed processing*, vol. 1, The MIT Press, 1986
- [15] C. Alippi, "Fault Tolerance Analysis in Back-Propagation Neural Networks", *Research Note RM/91/50*, Dept. of Computer Science, University College London, 1991
- [16] R. Negrini, M. Sami, R. Stefanelli, *Fault-Tolerance through Reconfiguration in VLSI Processing Arrays*, The MIT Press, Cambridge, MA, 1989
- [17] P.Y. Alla, G. Dreyfus, J.D. Gascuel, A. Johannet, L. Personnaz, J. Roman, M. Weinfeld, "Silicon integration of learning algorithms and other auto-adaptive properties in a digital feedback neural network", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks*, Dortmund, Germany, 1990