Pre-print version

S. Demidenko, V. Piuri, "Concurrent diagnosis in digital implementations of neural networks", in *Neurocomputing*, Elsevier, pp. 879-903, October, 2002. ISSN: 0925-2312.
DOI: 10.1016/S0925-2312(01)00678-6

Concurrent diagnosis in digital implementations ofneural networks

Serge Demidenko[a;*], Vincenzo Piuri[b]

[a]Institute of Information Sciences and Technology, Riddet Complex, Private Bag 11 222, Palmerston North, NewZealand
[b]Department of Information Technologies, University of Milan via Bramante 65, 26013 Crema (CR), Italy

Received 7 October 2000; accepted 13 August 2001

Abstract

Diagnosis is a basic issue ofany fault-tolerance policy. Fault localization within the neural architecture is necessary to provide information for hardware recon2guration in order to achieve system survival, possibly with reduced computational capabilities. In this paper, a comprehensive approach to architectural fault-tolerant design of neural networks is proposed and evaluated, with speci2c reference to concurrent high-level diagnosis and fault localization. The approach refers to the operational life of trained neural networks. Two error detection techniques are applied: on-line concurrent diagnosis with the use ofdata coding for error detection at neuron level and on-line compact testing for localization of the faulty neuron within the network.

Keywords: Digital implementation; Fault tolerance; Concurrent diagnosis; Signature analysis; Data coding

1. Introduction

The use ofarti2cial neural networks (ANNs) in mission- and life-critical applications implies adoption ofsuitable techniques at architectural level to provide fault tolerance during the operational life, in particular when maintenance is di9cult or impossible, or when high availability is mandatory. The importance of fault

* Corresponding author. Tel.: +64-06-350-5799-2457; fax: +64-06-350-5604.
E-mail address: s.demidenko@massey.ac.nz (S. Demidenko).

tolerance in many application areas (e.g., industrial applications, communication, avionics, autonics, and biomedics) during operational use has been made clear, e.g., in [17]. Relying only on the possible intrinsic fault-tolerance ability of the neural computation was shown in fact to be not suitable and safe [25,7,2]. Concurrent error detection allows to guarantee continuous checking ofresults; concurrent diagnosis is required to localize the faulty components as fast as possible in order to minimize the time for system repairing. System survival can then be achieved by excluding the faulty components from the active computation (e.g., by means ofrecon2guration), ifenough redundancy has been provided.

In this paper, we propose a comprehensive approach to fault tolerance in ANNs by using concurrent error detection techniques to identify (at neuron level) the presence of errors due to faults, on-line testing techniques to localize the faulty element in the network, and structural recon2guration to repair—ifpossible—the system. We consider these issues during the operational life of fully trained neural networks in order to satisfy the fault tolerance needs of the applications. We focus our attention on digital implementations of multi-layered feed-forward networks [15], since they have wide application areas (in particular, in signal and image processing). A functional error model—which has been proved eDective in a number ofdigital architectures and implementations—can be de2ned to describe the erroneous computation due to fault occurrence [25,7,2,26]. It is worth noting that errors due to faults in the neural architecture are diDerent from learning or recall errors which are intrinsic in the neural computation and related to the neural generalization ability; these last kind oferrors must not be considered from the point ofview of fault tolerance since they are a natural part ofthe neural behavior.

Concurrent error detection may be performed by means of traditional data coding techniques, suited to the speci2c neuron implementation [7,26,27]: neural operations are performed in the codeword space and, then, results are checked to detect the error occurrence. Localization of faulty neurons is performed by using on-line compact testing techniques [21,26] (in particular, signature compression [14,6]), through the following two main steps: identi2cation of the layer containing the faulty neuron, and localization of the faulty neuron within such a layer. The 2rst step may be realized by means ofrather modest built-in facilities, while the second one may be performed by a network's external observer to minimize the silicon area overhead. Finally, recon2guration ofthe interconnections among neurons is adopted in redundant modular architectures [13,20] to exclude the faulty neuron from the active computation and restore the nominal computational paradigm, if enough spare units are provided.

Section 2 summarizes the description ofthe neural computation, the error model for the class of architectures, and the computational paradigms considered in this paper. In Section 3, we present the general scheme for concurrent diagnosis: high regularity and modularity are considered to guarantee physical realizability and scalability. Section 4 summarizes the use ofdata coding to achieve concurrent error detection at neuron level: this step is necessary to supply the information for error localization. One ofthe main problems addressed by this paper is localization ofthe faulty components; it is discussed in detail in Section 5. At 2rst, the

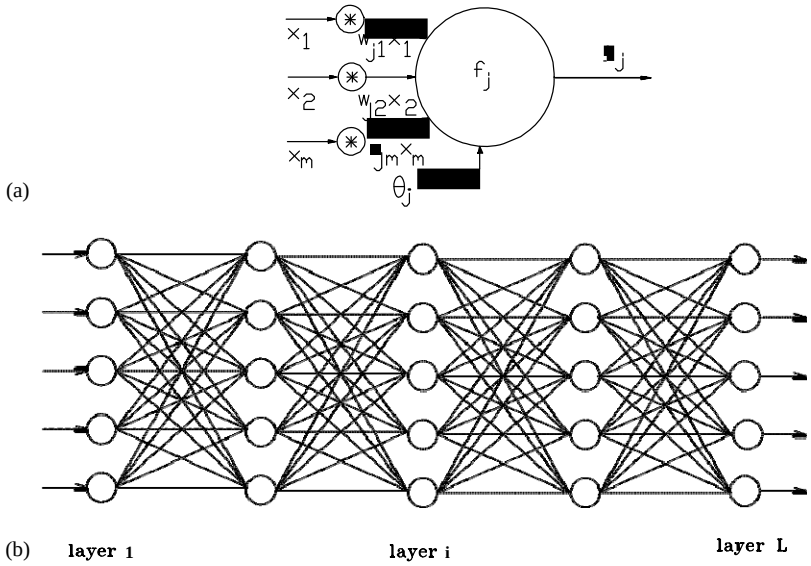

Fig. 1. A multi-layered feed-forward neural network: the neuron model (a); and the network structure (b).

theoretical bases ofour approach to the use ofsignature compression are given; then, application to the localization problem, architectural details, and evaluations (hardware overhead and computational delay) ofthe additional components are analyzed. Policies and hardware supports for possible recon2guration are 2nally considered in Section 6 to show the use ofthe information generated by our approach to error localization.

2. Neural and error models

A number ofcomputational paradigms have been proposed in the literature. In this paper, we consider the widely adopted neuron's model [15] (see Fig. 1) de2ned by

$$Y_j = f_j(\ _j - \#_j) = f_j \Big(\sum_i W_{ji} X_i - \#_j \Big) \ ; \qquad (1)$$

where X_i are the inputs, Y_j is the output, f_j is the non-linear activation function, W_{ji} are the synaptic interconnection weights, and $\#_j$ is the threshold. Our analysis has been performed on multi-layered feed-forward networks composed by such neurons: the basic results may be extended to other classes ofneural paradigms which can be built by using similar components.

Direct digital implementation ofEq. (1) involves one-to-one mapping onto digital components: weights are stored in memory elements, synaptic products are

evaluated by digital multipliers, summation is performed by a multiple-input adder, and a suitable circuit computes the non-linear function. For simplicity's sake, we restrict our analysis to 2xed-point arithmetic units.

In our analysis we consider only the case ofcompletely trained ANNs, although the basic ideas can be extended to deal with architectures capable ofon-board learning.

The fault model we adopt is based on the general digital structure introduced above [26]: possible faults may occur within any ofthe components ofthe neural architecture (weight storage, arithmetic devices, non-linear function evaluator).

A functional error model is de2ned to describe the erroneous neural computation induced by the occurrence of faults in the ANN's architecture, independently ofany technological implementation; our model has been proved eDective in a number of digital architectures and implementations [25,7,2,26,27]. Errors taken into account can belong to any ofthe subsequent classes:

- synaptic errors: de2ned as producing an unexpected value ofthe weight-signal product. These errors may be due to an error aDecting either the synaptic weight or the associated multiplication; in the absence ofany implementation detail, these two instances must be considered as indistinguishable.
- summation errors: they aDect the summation ofthe weighted inputs within the individual neuron;
- errors aDecting the non-linear evaluation function within the individual neuron; no particular assumption is introduced concerning the form of the erroneous function. Obviously this implies an erroneous value forwarded to all neurons connected to such neuron's output.

In the functional error model, the single error is any erroneous neural computation (induced by hardware faults) which aDects only the output of one neuron. For simplicity's sake, we consider only the case ofsingle error, except where otherwise explicitly noted.

In the layers following the faulty one, the single neuron error may induce multiple errors, i.e., several neurons' outputs may be erroneous due to the multiple propagation ofthe erroneous data. Ifa fault-free neuron receives erroneous inputs and generates an erroneous output according to the nominal speci2cation ofits behavior, it should not be considered faulty, even if its output has not the value corresponding to the correct inputs. Aliasing may occur due to the actual values of interconnection weights and to the presence ofreconvergent interconnection paths.

3. Concurrent diagnosis: the general scheme

ANN diagnosis simultaneously with the nominal computation can be achieved by applying the following operations:

1. detection ofthe error presence within neurons due to a hardware fault,

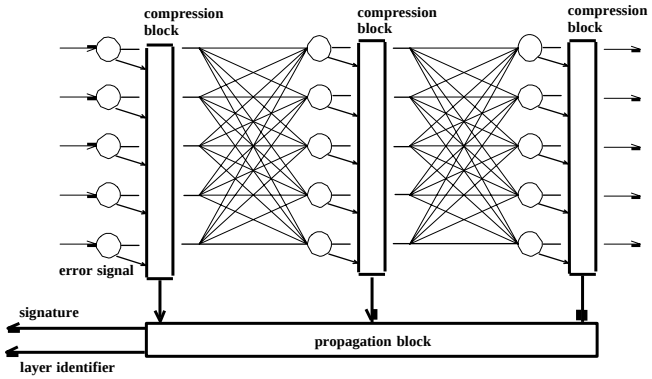

Fig. 2. The general structure for concurrent diagnosis.

2. identi2cation ofthe layer containing the faulty neuron, and
3. localization ofthe faulty neuron within such a layer.

The speci2c architectural solutions are strictly related to the class ofANNs and, thus, to the characteristics oferror propagation through the neural computation.

The basic scheme for implementing this approach is shown in Fig. 2. Each neuron includes the architectural supports for concurrent error detection. Whenever a fault induces an error in the neural computation inside neurons, the error signal generated in the faulty neuron becomes active. Data which identify the faulty layer and the faulty neuron within such a layer are generated by a diagnostic-data compacting block (based on data compression) inside each layer, and by a diagnostic-data propagating block which is shared by all layers.

For the neurons belonging to a layer, a short diagnostic word (signature) is generated by compressing the information associated with the neurons' error signals. In the single-error assumption, when a fault occurs and is detected within neurons, an erroneous signature (i.e., a signature diDerent from the reference value, corresponding to the fault-free network) is generated for the layer containing the faulty neuron.

Due to the multiple distribution oferroneous data in subsequent layers, also the signatures computed for such layers may be erroneous if the neurons' nominal outputs are not re-encoded before being delivered to the subsequent neurons. If aliasing is induced by the actual interconnection weights or by reconvergent paths and no re-encoding is adopted at the neurons' outputs, the layer's signature is correct even ifan error has been detected in at least one ofthe previous layers; since the neural computation proceeds from the input layer towards to the output one, the faulty neuron belongs to the layer which has an erroneous signature and which is the nearest to the input layer.

Propagation ofthe erroneous signature induced by the faulty neuron is realized by means ofthe propagation block. Towards the external environment, it delivers the erroneous signature and the identi2er ofthe layer containing the faulty neuron.

Extension to the multiple case can be obtained by sequential delivering ofthe above information concerning each faulty neuron.

4. Concurrent error detection

The neuron's architecture can be enhanced to support concurrent error detection. Modular redundancy with outputs' comparison is conceptually very simple but has an impressive cost in terms ofadditional silicon area; some algorithmic approaches can be considered [24], but they are often too application-speci2c.

Some data coding techniques give attractive results, without aDecting the computational performance in a signi2cant way [7,26,27]. Besides, they do not require any modi2cation ofthe neural computation, i.e., they do not aDect the neural characteristics ofthe network (learning, recall, generalization) with respect to the theoretical de2nitions. Focusing on AN codes [26,28] and on residue codes [27,28] is justi2ed since they are cost-eDective as far as structure redundancy and detection capability are concerned. They support our fault models both for arithmetic units and for the other components (in particular, storage units and function evaluators).

4.1. AN codes

AN codes are non-systematic codes in which each nominal datum N is replaced by its coded representation $C(N)$ via the linear transformation $C(N) = A \times N$, where the integer constant A is the code generator [26,28]. Arithmetic units (adders, subtractors, multipliers) are not modi2ed by the adoption ofthe coded data; the only diDerence with respect to the nominal structure is given by the number of bits required, that obviously increases with A.

The coding unit is a simple multiplier; since A is a constant, its structure can be optimized to grant maximum compactness and speed. Decoding is performed, for addition and subtraction involving coded operands, by applying the linear anti-transformation $N = C(N)/A$; the same decoding holds for multiplication whenever one operand only is in coded form. If both operands are coded, for a multiplication the antitransformation becomes $N = C(N)/A^2$. The structure ofthe neuron with concurrent detection capabilities by using AN codes is shown in Fig. 3; from the coded weighted input summation, the non-linear function $\mathbf{f}^*$ generates the coded output corresponding to the output generated by the function $\mathbf{f}$ in the AN code.

Whenever the antitransformation produces a non-null residue, an error is detected. Aliasing is possible whenever the error is a multiple ofthe code generator. It has been proved that $A = 3$ gives satisfactory detection capacity for all arithmetic units in which a single fault induces any single error that can be represented as $\pm 2^k$ [28].

AN + B codes are an extension ofthese codes that were introduced to improve the detection capabilities by removing the null codeword. Arithmetic operations and architectures are similar to the previous ones, even ifadditional circuits must be considered to deal with the code displacement B [26].

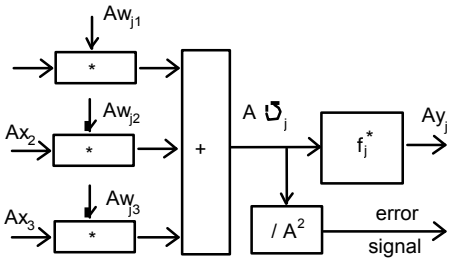

Fig. 3. The neuron with concurrent detection capabilities: the use ofAN codes.

Both these classes ofcodes are capable ofdetecting all errors in synaptic multipliers and in adders performing the input summations, ifa proper design ofthese components is adopted [26]. All single errors in the weight memory may be detected ifthe weights are coded by using any odd A; for AN codes, possible latency may occur ifthe corresponding neuron's input is divisible by A. Ifneuron's inputs are coded, single stuck-at faults on transmission lines of interconnection paths between pairs ofneurons can be detected; for AN codes, aliasing occurs ifweights associated to the faulty interconnections are divisible by A. If both inputs and weights are coded in the AN + B code, all single errors in arithmetic units, in weight storage, and in interconnection paths are detected without aliasing. As the evaluator ofthe non-linear function is concerned, error detection is strictly related to the adopted implementation technique: due to the non-linearity, AN codes cannot usually be exploited to provide this feature (see [26] for a detailed discussion).

4.2. Residue codes

In residue codes [27,28], each nominal datum N constitutes the information bits ofthe codeword, and the redundancy bits are given by the residue $R_b(N)$ of the datum N modulo b, where b is the base ofthe code. The coded representation $C(N)$ is obtained by using the bit cascade $C(N) = [N \mid R_b(N)]$. An accurate choice ofthe code base must be considered to achieve the highest detection capability at the minimum cost; a number ofpapers were published on the optimum choice of b to minimize aliasing (e.g., [28,23]). It has been proved that $b = 3$ gives complete detection capability for all arithmetic units in which a single fault induces an additive error [28], as in the previous class ofcodes.

Residue codes are separate since the computation on the nominal datum can be performed in parallel to computation on the redundancy bits: they have to be completed before result checking. The arithmetic units—adders, subtractors, multipliers—computing the nominal operations on the information bits are not modi2ed by the adoption ofthe coding scheme. The arithmetic units working on the redundancy bits must be capable ofoperating in the modulo arithmetic [13]. The coding unit must compute the residue modulo b ofthe uncoded data, while decoding is not required since the nominal output can be directly extracted from the

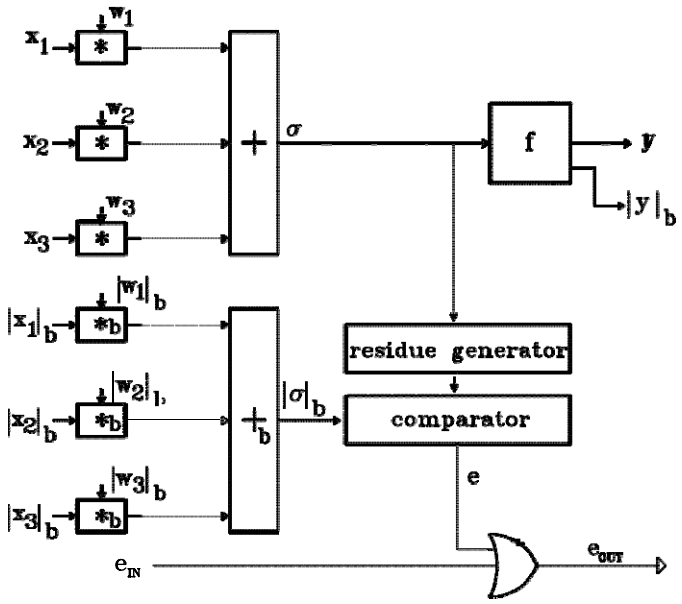

Fig. 4. The neuron with concurrent detection capabilities: the case ofresidue codes.

output codeword by dropping oD the redundancy bits. The neuron's structure by using residue codes is shown in Fig. 4.

Result checking consists ofcomparing the residue ofthe information bits of the nominal output to the redundancy bits generated from the redundancy bits of the nominal input operands. This operation is performed by using a coding unit (to obtain the residue ofthe information bits) and a comparator. Whenever the comparator inputs are diDerent, an error is detected. Aliasing occurs ifthe residue modulo b ofthe error is null.

The previous classes ofcodes have the same detection capabilities with respect to the considered error model introduced in Section 2. As far as the silicon area occupied by the additional circuits is concerned, experimental implementations have shown that the use ofresidue codes requires an area which is slightly higher than in the case of AN codes application due to the modulo operations. Conversely, the computational time has an opposite trend since AN codes involve operation on longer operands than residue codes. The choice ofthe solution which is the best suited to a given application must therefore be performed by considering the actual architectural and implementation technologies, i.e., the actual evaluation of the design parameters.

5. Concurrent localization

The localization ofthe faulty component in the neural network may be performed concurrently with the nominal operations whenever the application is so

critical as to require an immediate recon2guration ofthe system for guaranteeing the maximum availability. DiDerent techniques can be adopted to achieve this goal. In [26], a triangularization approach is proposed for highly regular feed-forward networks implemented by means ofa modular array structure; however, this solution may present some drawbacks concerning the regularity and modularity ofthe structure for networks composed by diDerent-sized layers. The conventional n-to-m binary encoder can be adopted for small layers, with limited routing.

The use ofon-line compact testing is more general and provides a structured approach to error localization and, thus, to identi2cation ofthe faulty components in large neural networks, while preserving the regularity and modularity ofthe architecture and limiting the additional wiring and circuit complexity. Compact testing is usually described as testing with compressed data [21,16]. In the literature, several data compression techniques (i.e., procedures for mapping binary words, binary sequences, or sequences ofbinary words onto a compact representation) have been presented, not only in diagnostics [21,29] but also in digital communications [8,19], in reliable computing [32,33], and in other areas; a uni2ed view is presented in [31].

There are some speci2c and partially conJicting requirements in choosing the compression technique which is the most suited for a given application case. First ofall, the compaction degree must be quite high and the complexity ofthe compaction algorithm must be reasonably modest in order to minimize the diagnostic hardware and the time overhead, and to simplify the comparison between the actual compression result and the reference value (corresponding to the fault-free operation). On the other hand, compression should not lead to any error masking with respect to the adopted error model (i.e., in our assumptions, any single error should be uniquely identi2ed by the compact representation computed on the actual data). Besides, ifthe presence ofan error is detected, the information contained in the compact representation should guarantee a correct localization ofthe erroneous datum belonging to the sequence which has been compressed (and, consequently, ofthe faulty component).

One ofthe methods that satis2es the above requirements is based on data compression by using linear feedback shift registers; its application for digital testing is widely known as signature analysis [14,6]. In this section, the theoretical basis for signature analysis is given and diDerent approaches for its application to error localization in ANNs are discussed.

5.1. Basic remarks on signature analysis

The essence ofsignature analysis is based on compression ofthe output sequence ofthe circuit under test into a short word (called signature) by using a linear feedback shift register (the signature register), and on comparison ofthe signature obtained from the actual outputs with the reference value (corresponding to the faulty-free operation). If the actual signature matches the reference value, the circuit under test is assumed fault free (presuming that the input test is correct and consistent); otherwise, it is considered faulty [14,6].

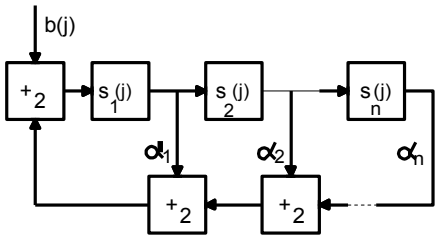

Fig. 5. Single-channel signature register.

In the literature, two kinds ofapproaches have been proposed according to the number ofinput streams entering into the register at each operation step: the single-channel signature registers (SCSR) and the multi-channel ones (MCSR). In this section, we introduce a general analytical description for data compression operated by these registers. Let the following notations be used:

- n is the length ofthe signature register, i.e., the highest degree ofthe characteristic polynomial ofthe signature register [34,22]; it is $n \in \{1; 2; 3; : : :\}$;
- a_i is the ith coe9cient ofthe characteristic polynomial; it is $a_i \in \{0; 1\}$; $i \in \{1; 2; 3; : : : ; n\}$;
- $b_i(k)$ is the bit value ofthe ith sequence being compressed that penetrates into the i-th input ofthe signature register at the kth time step, $k = 0; 1; 2; 3; : : :$;
- $s_i(k)$ is the state (i.e., the value) ofthe ith bit ofsignature register at the kth time step.

Consider 2rst the SCSR; without loss ofgenerality, we take into account only SCSR having the input into the 2rst bit ofthe register. Fig. 5 illustrates its block diagram, while its operation is de2ned by [10]

$$s_i(j) = \left[\bigoplus_{m=1}^{n} s_m(0) \bigoplus_{t=m}^{n} a_t C_{(j+m-t-i)} \right. \oplus \left[\bigoplus_{z=0}^{j-2} b_1(z) \bigoplus_{\beta=1}^{n} a_\beta C_{(j-\beta-z-i)} \oplus b_1(j-i) \right]; \quad j \geq i; \ i = 1; 2; 3; : : : ; n; \tag{2.1}$$

$$s_i(j) = s_{i-j}(0); \quad j < i; \ i = 1; 2; 3; : : : ; n; \tag{2.2}$$

where $s_i(0)$ is initial state ofthe ith bit ofSCSR, $\oplus$ denotes the modulo-2 summation, and $C_{(w)}$ is the sum ofproducts ofthe characteristic polynomial coe9cients a_i ofthe signature register.

The set ofsubscripts ofthe characteristic polynomial's coe9cients in each product is a unique variant ofthe restricted composition ofthe number w, in which

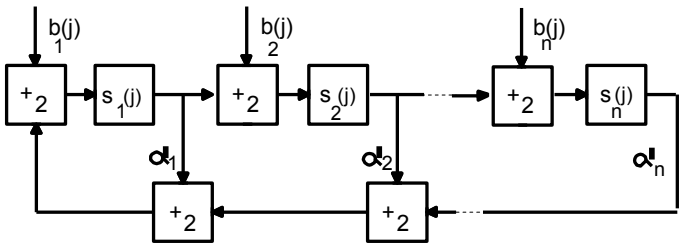

Fig. 6. Multiple-channel signature register.

only integers not greater than n are used [30]. Therefore, it is

$$C_{(w)} = \begin{cases} 0; & w < 0; \\ 1; & w = 0; \\ \bigoplus_{d=1}^{D_{(w)}} \prod_{q=1}^{N_{(w;d)}} a_{p_{(d;q)}}; & w > 0; \end{cases} \tag{3}$$

where $\sum_{q=1}^{N_{(w;d)}} p_{(d;q)} = w \ \forall d \in \{1;2;3;\dots;D_{(w)}\}$; $p_{(d;q)} \in \{1;2;3;\dots;n\}$;

$$\{p_{(g;q)} | q = \overline{1;N_{(w;g)}}\} \neq \{p_{(h;q)} | q = \overline{1;N_{(w;h)}}\} \quad \forall g \neq h;\ g;h \in \{1;2;3;\dots;D_{(w)}\}.$$

The set ofsubscripts $p_{(d;q)}$ for each value of d in expression (3) is a unique variant ofcomposition ofthe number w. The summation limit $D_{(w)}$ equals the number of all these compositions [30], while $N_{(d;w)}$ equals the number ofsummation terms of the dth variant ofcomposition.

Expression (2.1) consists oftwo parts. The 2rst one is dependent on the initial state ofthe register only, while the second part is dependent on values ofthe elements ofthe sequence being compressed. In other words, the 2rst part is a description ofa pseudorandom sequence generator [34]. This completely agrees with the well-known fact that the operation of a signature register without input information (i.e., with null input) corresponds to a binary sequence generation having maximum period equal to $2^n - 1$ (ifthe characteristic polynomial is primitive and irreducible). The second part ofexpression (2.1) corresponds to a single-channel signature register having null initial state.

The MCSR is widely used to test multi-output circuits. Fig. 6 presents its block diagram, while operation is de2ned by [10]

$$s_i(j) = \bigoplus_{m=1}^{n} s_m(0) \bigoplus_{t=m}^{n} a_t C_{(j+m-t-i)} \oplus \left[\bigoplus_{z=0}^{j-2} \bigoplus_{t=1}^{n} b_t(z) \bigoplus_{f=t}^{n} a_f C_{(j+t-f-z-i-1)} \oplus \bigoplus_{r=1}^{i} b_r(j-i+r-1) \right]; \quad j > i;\ i = 1;2;3;\dots;n; \tag{4.1}$$

$$s_i(j) = [s_{i-j}(0)] \oplus \left[\sum_{r=i-j+1}^{i} \oplus b_r(j - i + r - 1) \right]; \quad j \geq i; \quad i = 1; 2; 3; \ldots; n. \tag{4.2}$$

Expression (4.1) also consists oftwo parts: the 2rst one describes a pseudo-random sequence generator, the second one describes the multi-channel signature register having null initial state. In comparison with (2.1), the second part of(4.1) includes an additional summation over the variable t. This reJects the fact that n inputs are used for n-channel compression in the signature register.

In our approach to error localization within each layer, we consider only the SCSR solution since we assume that error bits generated by the neurons (one error bit per neuron) constitute the input stream being compressed and that only one of such bits is compressed at each operation step ofthe register. This is allowed by the single-error model we adopted. In the case ofmore complex models and implementation architectures (e.g., when multiplexing is concerned), we should consider also the use ofMCSR; however, this solution often requires wider redundant circuits, even ifit does not modify the general schemes discussed here. For simplicity, but without loss in generality as the comprehensive approach is concerned, only SCSR will be considered in the subsequent analysis.

The neural network is considered faulty whenever the actual signature S diDers from the reference value S^* (corresponding to the fault-free circuit) [14,34]. The error pattern occurring is characterized by the error signature $S' = S \oplus S^*$. The error signals, produced by neurons belonging to a layer, are sent into the compression chain associated with such layer to produce the error signature. The error signal is 0 ifno error has been detected in the neural computation, 1 otherwise; the input sequence for the signature register is composed of0's only ifno error has been detected in the layer and the signature is null. Ifa fault induced an error in the output ofa neuron, at least one element ofthe sequence is 1; ifno aliasing masks the error, the actual signature is not null. Under the assumption ofsingle neural errors, exactly one element ofthe sequence is 1. The error signal associated with the whole layer can be obtained by OR-ing all bits ofthe layer's signature.

By applying the technique discussed above, we can 2nd which bit ofthe input sequence is erroneous (i.e., which bit is not null) and, thus, which neuron in the layer is faulty. In our case, the reference signature is null since the bits being compressed are the error bits generated by each neuron; the error signature is thus equal to the actual signature.

Several approaches have been proposed in the literature to locate the erroneous bits in a sequence being compressed by using the error signature [18,12,9,11]. A general analytical approach has been proposed in [10]: from this analysis, a practical methodology has been derived to generate all possible erroneous input patterns which correspond to the given error signature. We generate in fact a system oflinear Diophantine equations for the given value ofthe polynomial coe9cients and for the given time step. Each bit of the error signature is associated with the bits ofthe error sequence. Such a system is solved with respect to the set

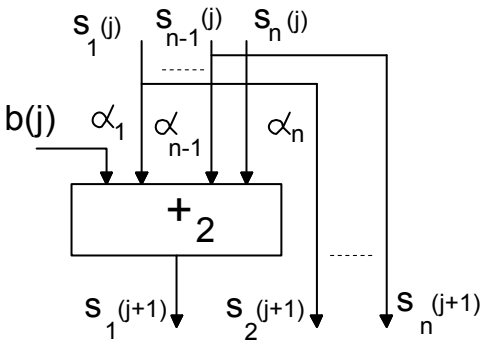

Fig. 7. The basic block ofthe distributed SCSR.

ofelements ofthe error sequence; solutions identify the position ofall possible erroneous bits, i.e. ofall possible faulty neurons.

Application ofone ofthese approaches to error localization in the case ofANN diagnostics is the main goal ofthe present paper. This choice is strongly supported by the well-known fact that signature analysis provides complete coverage of all single- and double-bit errors occurring in the output sequence ofthe circuit under test, while all other errors are covered approximately at $100(2^m - 1)=2^m\%$ [34]. Therefore, it is capable of satisfying the requirements for unique localization of the faulty neuron within a layer.

5.2. Application of the signature analysis: the linear signature generator

To preserve the regularity ofthe architecture and to avoid a complex sequential scheme, the signature generator can be realized by means ofa distributed SCSR architecture, which is functionally equivalent to the structure presented in Fig. 5. This architecture is composed ofa cascade ofmodulo-2 adders which are properly interconnected in a linear chain according to the SCSR computational scheme; index **j** represents the **j**th neuron ofthe considered layer, while data from the previous stage are weighted by using the coe9cients $a_1; a_2; : : : ; a_m$. The basic unit ofthe compression cascade is shown in Fig. 7.

The traditional scheme ofsignature compression recirculates the signature bits (see the loop in Fig. 5) to deal with a new input bit at each step. This design is related to the usual applications ofSCSR in which the inputs are supplied serially. In our ANN's architecture the input bits ofthe sequence that must be compressed are available in parallel. Therefore, we rearrange the SCSR computation in the distributed scheme, in which each computational step ofthe traditional algorithm is performed by a diDerent subset of hardware components. Such components are similar to the traditional one, but they are interconnected so that each stage incorporates the contribution ofthe corresponding error bit in the signature without any signature recirculation inside a stage. In other words, we apply an unrolling technique to the traditional scheme.

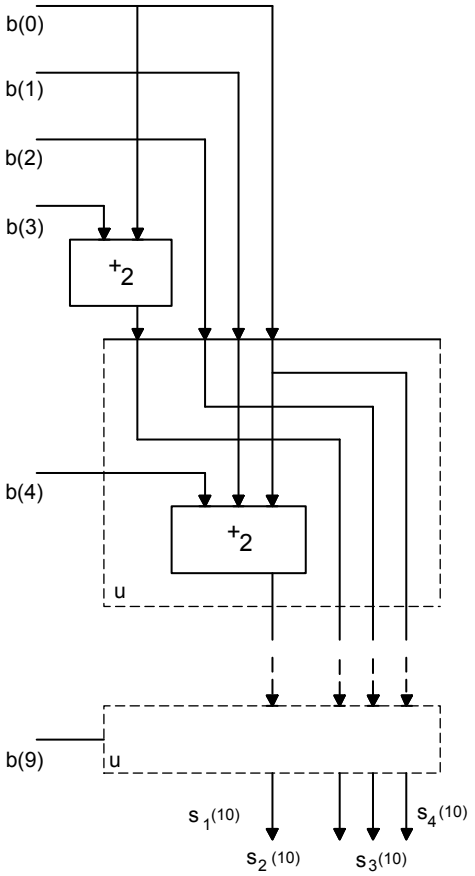

Fig. 8. The linear signature generator.

Fig. 8 presents an example ofa linear signature chain for 10 neurons: the polynomial coe9cients are $a_1 = a_2 = 0$, $a_3 = a_4 = 1$. The symbol u identi2es the basic unit ofthe compression cascade, which has been shown in Fig. 7; adders in the 2rst three units are not necessary since initial data for the signature compression are null.

5.3. Application of the signature analysis: the tree-like signature generator

To reduce the time required for signature generation, we propose a tree-like organization ofthe basic blocks in order to exploit the intrinsic computational parallelism ofthe tree-like scheme. The linear chain ofthe previous section is partitioned into k sub-chains; each ofthem contains m=k modulo-2 adders. A simple example for a two-level tree-like structure is presented in Fig. 9: the number of neurons in the layer and characteristic polynomial are the same as considered

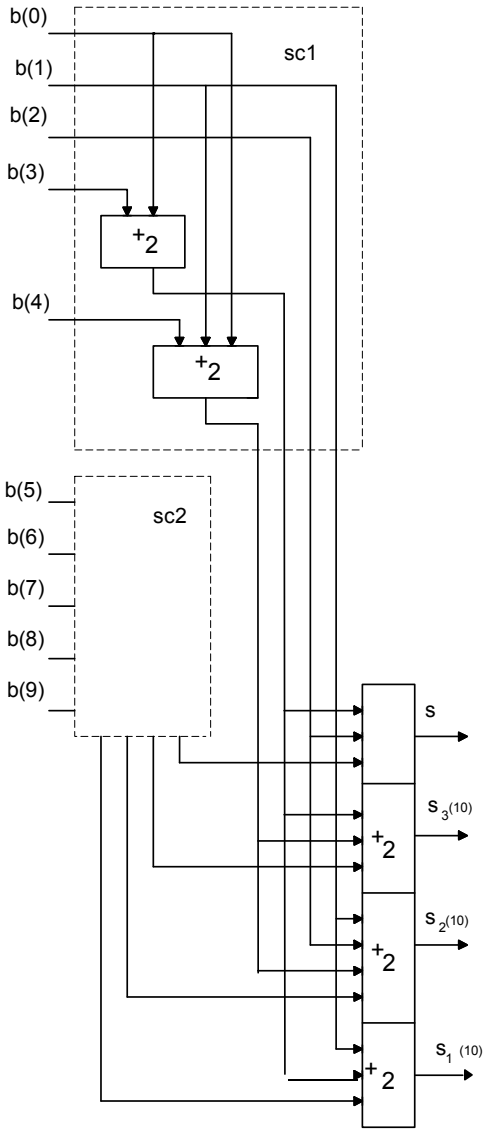

Fig. 9. The tree-like signature generator: the two-level case.

for the example ofFig. 8. The chain ofmodulo-2 adders is partitioned into two sub-chains (sc1 and sc2): their structure is similar to those discussed for the linear organization.

The computation ofthe sub-chains' signatures (the intermediate signatures) is performed simultaneously; the intermediate signatures are then merged to generate the 2nal layer's signature. Interconnection between the sub-chains can be achieved

by applying the following remarks. In the linear structure, the initial state of the signature compression for any segment of the chain is the result generated by the signature compression in the previous segment. In Section 5.1, we have shown that the signature compression contains two terms: the pseudo-random generation with the nominal input data and the signature compression with null initial state. These terms can be computed in parallel. We compute the components oftwo intermediate signatures by considering the initial null state; then, we generate the pseudo-random value after k steps (where k is the number ofneurons in the second sub-chain) by using the 2rst computed intermediate signature as the initial value for the pseudo-random process. Finally, we add (modulo-2) this value to the second intermediate signature to generate the layer's signature.

From the expressions discussed in Section 5.1, we derived an algorithm for pseudo-random generation without implementing directly the pseudo-random process [10,11]. For the adopted coe9cients $a_1; a_2; : : : ; a_m$ and for the given **j**, we compute the compositions and we obtain a set ofBoolean equations which relate the initial state ofthe pseudo-random generator to its value after the **j**th step. For our example, we have

$$s_1(5) = s_2(0) \oplus s_4(0);$$

$$s_2(5) = s_1(0) \oplus s_3(0) \oplus s_4(0);$$

$$s_3(5) = s_1(0) \oplus s_2(0);$$

$$s_4(5) = s_2(0) \oplus s_3(0):$$

A more complex partitioning is shown in Fig. 10 for the case of eight neurons in the layer and for the characteristic polynomial having $a_1 = a_2 = 0$, $a_3 = a_4 = 1$; the blocks g provide interconnection ofsub-chains.

5.4. Application of the signature analysis: the signature propagator

Identi2cation of the layer containing the faulty neuron can be performed by analyzing the complete set ofsignatures generated by all layers. The signatures are tested starting from the one produced by the input layer and proceeding towards the output layer. Ifno re-encoding ofthe neurons' outputs is adopted, the faulty layer corresponds to the one having non-null signature; in the case ofmultiple faults, only the nearest to the primary inputs can be localized certainly, being the error signal ofthe subsequent neurons possibly active since either the neuron itself is faulty or the erroneous inputs generated an erroneous output. Conversely, if neurons' outputs are re-encoded before being delivered to the subsequent neurons, all faulty neurons can be localized, one layer at a time, since the error signal produced by each neuron is independent from the status of the other neurons.

This approach is practically realizable only ifthe total number ofbits for all signatures is reasonably low. Otherwise, when the number ofsignature bits is too high while the number oflayers is low, we compress the information allowing

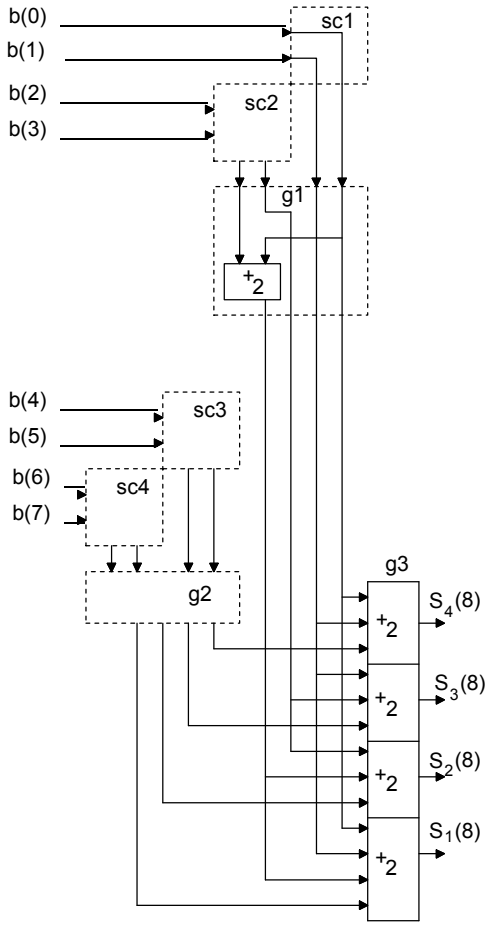

Fig. 10. The tree-like signature generator: the multiple-level case.

identi2cation ofthe faulty layer into the layer error signals: each ofthem is obtained by OR-ing all signature bits at the output ofthe corresponding signature generator. These signals are extracted from the architecture through separate lines. They are tested starting from the one produced by the input layer and proceeding towards the output layer: the faulty layer is identi2ed by the 2rst signal which is recognized active. The signatures are multiplexed on the same output lines and delivered to the signature analyzer for localization of the faulty neuron within the layer.

Ifthe number oflayers is too high, compression must be improved to reduce the number oflayer error signals. Each layer is identi2ed by its layer address, a unique binary combination; the identi2er ofthe faulty layer is propagated in parallel with the fault signature. The basic block of the subsystem for signature propagation is shown in Fig. 11. It contains a multiplexer for selecting the signature and the

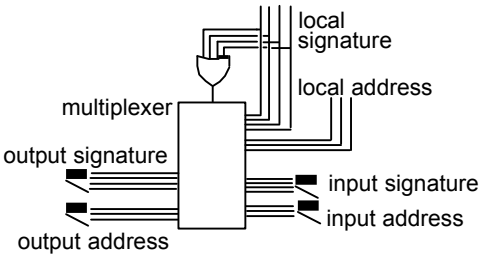

Fig. 11. The signature propagator block.

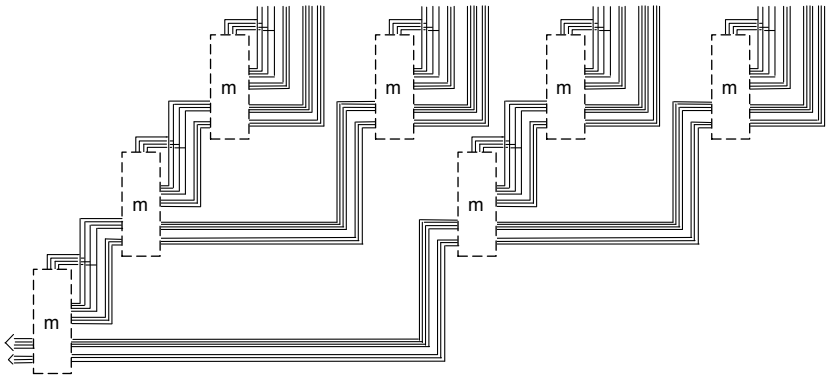

Fig. 12. The tree-like signature propagator.

layer address that must be propagated backwards to the ANN's input layer. Ifthe control signal is 1, the signature generated in that layer and the local address are propagated to the previous layer; otherwise, the signature and the address received from the subsequent layer are forwarded to the previous one. The multiplexer control signal is the layer error signal (it is 0 ifno signature bit ofsuch layer is 1). In the output layer, no multiplexing is required. The above structure ofthe signature propagator and the direction of the computational Jow in a feed-forward neural network guarantee that the faulty layer is correctly and uniquely identi2ed at the propagator's output: the faulty neuron is then uniquely identi2ed within such a layer by the error signature delivered by the signature propagator. Sequencing of the signature propagation can be considered to allow detection ofcontemporaneous faults in two or more layers when neurons' outputs are re-encoded.

To reduce the computational delay, we adopt a tree-like organization also for the signature propagator. The chain ofpropagator's modules is partitioned into k sub-chains, which are properly interconnected to guarantee extraction ofthe signature ofthe faulty layer; an example is shown in Fig. 12 (m is the module ofFig. 11). The module propagates the local signature or the signature coming from the subsequent layer (and the corresponding layer address) according with the multiplexer control signal, as in the previous case. No multiplexer is required

for the last layer ofeach sub-chain. The outputs ofthe sub-chains are then treated as they were outputs ofsignature generators.

Ifthe neuron's output is re-encoded before being delivered to the subsequent neurons, i.e., the output codeword is re-computed from the nominal output, each layer receives codewords from the previous one. As a consequence, there is never error masking due to path reconvergence. The syndrome ofeach layer account therefore only for the errors due to faults in that layer.

Propagation ofall the layers' signatures toward the observation and analysis point can be accomplished by applying the compression technique to the set of these syndrome.

5.5. Evaluation of the use of signatures: hardware and time overheads

In all architecture based on the structure presented in Fig. 2, the hardware overhead is mainly due to the circuits for concurrent error detection within each neuron and to the circuits for signature generation and propagation through the whole network. In particular, the overhead for concurrent error localization in the case of signature is related to the modulo-2 adders required to generate the signature, to the components allowing signature propagation, and to the additional interconnection paths associated to all the previous devices.

First ofall, we evaluate the silicon overhead due to the compression blocks; to simplify the analysis, we focus our attention on a single layer: results are valid for each layer ofthe neural network when the corresponding characteristic parameters are used.

In the case ofthe linear signature generator, since the input value ofthe 2rst modulo-2 adder ofthe chain is null, the number ofmodulo-2 adders is at most $m-1$, where m is the number ofneurons in the considered layer. We can reduce the number ofadders and number ofinputs ofeach adder by adopting a suitable choice for the signature characteristic polynomial. An interesting solution uses a minimum polynomial, i.e., a polynomial ofthe form $X^n \oplus X^q \oplus 1;\ q \in \{1;2;3;\ldots;n-1\}$, where n is the signature register length; each modulo-2 adder is a three-input device and the number ofadders is $m-q$. Good values of q are as close as possible to n. The optimum value holds for $q = n-1$; we need $m-n+1$ adders. The number ofinterconnections between the adjacent neurons within a layer is constant along the layer and it is equal to n.

In the case ofthe tree-like organization, each ofthe k sub-chains contains $m/k-1$ modulo-2 adders (we assume that m is divisible by k). For the optimum choice ofthe characteristic polynomial, we have $m/k-n+1$ adders in each sub-chain. Composition ofeach two intermediate signatures generated by the sub-chains need n additional adders: the total amount ofredundant adders is $m-(n-2)k-1$, by assuming that k is a positive power of2. The number ofinterconnection lines to realize compression is not constant due to the additional connections between the sub-chains. Inside each layer, there are n lines at the beginning and $n(\log_2 k+1)$ at the end ofthe structure. Suitable placement ofthe sub-chains minimizes the additional routing between the sub-chains.

The hardware redundancy due to the propagation block is related to the architectural solution adopted. In the case ofa linear propagation structure, it requires $L-1$ two-input multiplexers and $L-1$ OR gates to generate the multiplexer control signals. In the case ofa tree-like organization, the hardware overhead for the active devices is identical to the linear case, while interconnections may require a larger silicon area.

The second 2gure ofmerit that we have to take into account to select the approach which is best suited to a speci2c application is the computational delay r_d introduced by concurrent diagnosis with respect to the nominal neural computation. It is

$$r_d = \max_{i=1}^{L}(r_{Si}) + r_P - r_N = \max_{i=1}^{L}\left(\sum_{h=1}^{i} r_{Nh} + r_{Gi}\right) + r_P - \sum_{h=1}^{L} r_{Nh}; \qquad (5)$$

where r_{Si} is the computational time required to achieve steady values at the output ofthe signature generator for the ith layer after presentation of ANN's inputs, r_P is the computational time ofthe signature propagator, r_N is the computational time ofthe nominal network, r_{Nh} is the computational time oflayer h, and r_{Gh} is the computational time ofthe signature generator.

The computational time r_G due to the linear signature generator within a layer is proportional to the number ofsummation stages in the chain, i.e., to the number $m-q$ ofmodulo-2 adders. For the optimum choice ofthe characteristic polynomial, it is $m-n+1$ times the computational time ofan adder.

The computational time for the tree-like generator is due to the summation stages ofa sub-chain and to the circuits for sub-signature composition. The contribution ofthe sub-chains is proportional to $m=k-n+1$ for the optimum polynomial, while the contribution ofthe composition circuits is $\log_2 k$ times the computational time ofan adder. Therefore, the total delay is $m=k-n+1+\log_2 k$ times the computational time ofan adder.

The computational delay introduced by the propagation block is related to its structure. In the case ofthe linear propagator, the computational delay is due to the multiplexing stages ofthe propagation chain; in the worst case, the actual delay is equal to the time spent in the multiplexers ofthe whole propagation chain, i.e., it is $L-1$ times the delay introduced by a two-inputs multiplexer. In the case ofthe tree-like structure, by assuming a complete tree organization having $k=L=2$ blocks at the 2rst level, the computational time is $\log_2 L$.

6. The use of error localization information

Information generated by concurrent error detection, compressed by the compression blocks, and propagated by propagation blocks toward the network's outputs may be used to identify uniquely the faulty neurons and to provide system survival (possibly, at the nominal behavior) by excluding the faulty components from the active computation. Dynamic redundancy [20] may be in fact

implemented whenever high survival capability ofthe system is required, in particular for critical applications or when maintenance is di9cult or impossible.

To achieve the above goals, additional features must be introduced in the architecture on which the neural network is mapped. First ofall, a regular, recon2gurable architecture must be adopted: redundant neurons and interconnections must be added to the basic structure as spare components that can be used to replace the faulty ones. Whenever a neuron or an interconnection is recognized to be faulty, if enough redundancy has been provided, it is excluded from the nominal computation and it is replaced by a spare component. For example, ifa regular array structure with switched busses is adopted [13], replacement is achieved by recon2guring the interconnection switches [20].

Besides, monitoring the error information and recon2guration control must be accurately considered to achieve an e9cient and eDective system survival with respect to the application requirements. These activities are in fact necessary to perform the actual localization of the faulty neuron within the network and to generate the control signals for proper recon2guration of the interconnections among neurons. DiDerent approaches could be considered to implement these features by adopting suitable hardware supports.

The layer containing the faulty neuron is directly given by the layer address delivered by the propagation block. Localization ofthe faulty neuron implies analysis ofthe output signature delivered by the propagation block: the monitoring procedure must deduce the position of the faulty neuron within the layer from the actual value ofthe corresponding signature. A table storing such correspondence or an algorithmic approach (e.g., [11]) can be adopted to solve this problem.

On-board VLSI realization ofthe localization procedure may be considered for applications requiring high availability and short repairing time. OD-board (and possibly oD-line) solutions can be envisioned for very large networks or when only network credibility is critical; in this case, the outputs ofthe propagation block are extracted from the circuit implementing the network and delivered to the external observer.

Similarly, generation ofthe recon2guration control signals can be performed by adopting an on-board concurrent solution whenever high system availability is mandatory for the application envisioned. In this case, the recon2guration control subsystem uses the previous distribution of faulty neurons within the architecture underlying the neural network and the position ofthe new faulty neuron (generated by the monitoring subsystem) to identify a new mapping of the neural paradigm onto the hardware architecture. Ifsuch a mapping exists, the corresponding control signals are directly applied to the interconnection structure and the network is still able to continue its full operation; otherwise, network functions may be degraded till a completely erroneous behavior.

In the case ofcomplex interconnection structures, an eDective recon2guration policy cannot often be computed on-board by using a dedicated VLSI subsystem having reasonable dimensions. To deal with these cases and when very fast recon2guration is not mandatory, an oD-board approach can be adopted to se-

lect the suited mapping for the actual fault distribution. Control signals are then down-loaded onto the neural architecture to achieve, ifit is possible, the operating con2guration.

7. Concluding remarks

In this paper, a general comprehensive approach to fault-tolerant design of neural networks has been presented, with particular attention to the problems related to concurrent diagnosis. The basic structure ofthe digital network has been enhanced by introducing circuits for concurrent error detection based on arithmetic codes (AN codes or residue codes). Additional components for compressing the error signal generated within each layer have been de2ned by using signature analysis: in particular, the signature generator and the signature propagator have been discussed. Concurrent localization ofthe faulty neuron is 2nally achieved by observing the results oferror compression and propagation.

By using the information produced by our concurrent approach, it is possible to design an architecture capable ofon-board diagnosis and automatic recon2guration after fault detection and localization. This can be useful for minimizing the diagnosis time and the repairing time, and, thus, for maximizing system availability in highly critical applications. The approach proposed in this paper can be also directly adopted to protect pipelined architectures, when suitable pipeline registers are introduced also along the propagation block.

Further research will be performed to deal also with more complex error models than the one considered in this paper, although this last one has been shown eDective for many architectures. For example, an interesting case consists of multiple neural errors induced by multiplexing ofdigital components (multipliers, adders, non-linear evaluators). These error models have the same physical origin ofthe model adopted in this paper, but the possible multiple use ofdigital components can make error analysis and detection more di9cult. Appropriate techniques must be taken into (e.g., see [5,4]) to con2ne the multiple error propagation and avoid aliasing; they should be incorporated in the signature analysis by minimizing the aliases probability and the circuit complexity as well into a comprehensive design framework [1,3].

Acknowledgements

The authors would like to thank the National Research Council ofItaly, the Royal Society ofNew Zealand (ISAT Linkages Programme) and Politecnico di Milano for the sigini2cant support provided to the research.

The authors are grateful to the anonymous referees for providing comments and suggestions that greatly helped in improving this paper.

References

[1] C. Alippi, S. Ferrari, V. Piuri, M. Sami, F. Scotti, New trends in intelligent system design for embedded and measurement applications, IEEE Instrum. Measur. Mag. 2 (1999) 36–44.

[2] C. Alippi, F. Fummi, V. Piuri, M. Sami, D. Sciuto, Testability analysis and behavioural testing ofthe hop2eld neural paradigm, IEEE Trans. VLSI Systems 6 (3) (1998) 507–511.

[3] C. Alippi, V. Piuri, F. Scotti, Accuracy vs. complexity in RBF neural networks: a system level design approach, IEEE Instrum. Measur. Mag. 4 (2001).

[4] A. Antola, F. Ferrandi, V. Piuri, M. Sami, Semi-concurrent error detection in data paths, IEEE Trans. Comput. 50 (2001).

[5] A. Antola, V. Piuri, M. Sami, High-level synthesis ofdata paths with concurrent error detection, Proceedings ofthe IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems DFT'98, Austin, TX, USA, November 1998, pp. 292–300.

[6] P.H. Bardell, W.H. McAnney, J. Savir, Built-in-Test for VLSI, Wiley, New York, 1987.

[7] S. Bettola, V. Piuri, High performance digital neural networks: the use of redundant binary representation for concurrent error detection, IEEE Trans. Comput. 47 (3) (1998).

[8] R.E. Blahut, Theory and Practice ofError Control Codes, Addison-Wesley, Reading, MA, 1989.

[9] J.C. Chan, B.F. Womack, Diagnostics based on faulty signature, Proceedings of the International Test Conference ITC 89, September 1989, p. 935.

[10] S.N. Demidenko, V. Piuri, A generalized approach to error localisation by using fault signature information, Politecnico di Milano, Department of Electronics and Information, Report No. 009-93, Feb. 1993.

[11] S. Demidenko, V. Piuri, A. Ivaniukovich, Error localization in test outputs: a generalized analysis ofsignature compression, Proceedings ofthe Asian Test Symposium 93, Beijing, P.R. China, November 1993.

[12] O.N. Diachenko, A.N. Tarasenko, Modes of diagnostic information separation from signature, Electron. Model. 12 (5) (1990) 64–70 (Russian).

[13] F. Distante, M. Sami, R. Stefanelli, G. Storti-Gajani, Fault tolerance in massively parallel silicon architectures: the case ofneural nets, IEEE Proceedings, April 1991.

[14] R.A. Frohwerk, Signature analysis: a new digital 2eld service method, Hewlett-Packard J. 28 (1977) 2–8.

[15] J. Hertz, A. Krog, R.G. Palmer, Introduction to the Theory ofNeural Computation, Addison-Wesley, Reading, MA, 1991.

[16] S.K. Kazmina, Compact testing, Automat. Remote Control 43 (1982) 412–424.

[17] P. Lisboa, Industrial use ofsafety-related arti2cial neural networks, Report N. 327=2001, John Moores University, Liverpool, UK, 2001 (http:==www.hse.gov.uk=research=crr_pdf=2001=crr01327.pdf).

[18] W.N. McAnney, J. Savir, There is information in faulty signatures, Proceedings of the International Test Conference ITC 87, pp. 630–636, September 1987.

[19] A.N. Michelson, A.H. Levesque, Error Control Techniques for Digital Communications, Wiley, New York, 1985.

[20] R. Negrini, M. Sami, R. Stefanelli, Fault Tolerance Through Recon2guration in VLSI and WSI Arrays, MIT Press, Cambridge, MA, 1989.

[21] K.P. Parker, Compact testing: testing with compressed data, Proceedings ofthe Fault Tolerant Computing Symposium FTCS 6, 1976, pp. 93–98.

[22] W.W. Peterson, E.J. Weldon, Error Correcting Codes, MIT Press, Cambridge, 1972.

[23] V. Piuri, Fault-tolerant systolic arrays: an approach based upon residue arithmetic, IEEE Proceedings ofthe ARITH-8, Como, Italy, 1987.

[24] V. Piuri, An algorithmic approach to concurrent error detection in arti2cial neural networks, IEEE Proceedings ofthe International Workshop on VLSI Signal Processing, Napa, CA, USA, October 1992.

[25] V. Piuri, Analysis of fault tolerance in arti2cial neural networks, J. Parallel Distr. Computing 61 (2001) 18–48.

[26] V. Piuri, M. Sami, R. Stefanelli, Arithmetic codes for concurrent error detection in arti2cial neural networks: the case ofAN+B codes, IEEE Proceedings ofthe International Workshop on Defect and Fault Tolerance in VLSI Systems, Dallas, TX, USA, 1992.
[27] V. Piuri, M. Villa, Residue codes for concurrent error detection in arti2cial neural networks, Proceedings ofthe IJCNN93, Portland, OR, USA, July 1993.
[28] T.R.N. Rao, Error Coding for Arithmetic Processors, Academic Press, NY, 1974.
[29] S.M. Reddy, K.K. Saluja, M.G. Karpovsky, A data compression technique for built-in self testing, IEEE Trans. Comput. 37 (1988) 1151–1156.
[30] J. Riordan, An Introduction to Combinatorial Analysis, Princeton University Press, Princeton, 1980.
[31] J.P. Robinson, N.R. Saxena, A uni2ed view oftest compression methods, IEEE Trans. Comput. 36 (1987) 94–99.
[32] J.A. Storer, Data Compression: Methods and Theory, Computer Science Press, Rockwille, MD, 1988.
[33] J. Wakerly, Error Detection Codes, Self-Checking Circuits and Applications, North-Holland, Amsterdam, 1978.
[34] V.N. Yarmolik, S.N. Demidenko, Generation and Application ofPseudorandom Sequences for Random Testing, Wiley, Chichester, 1988.

Serge Demidenko received his MSc in Computer Engineering in 1977 from the Belarusian State University ofInformatics and Radio Electronics, Ph.D. in Engineering Cybernetics and Information Theory in 1984 from the Institute ofEngineering Cybernetics ofthe Belarusian Academy ofSciences, Associate Professor by Research (Senior Scientist) Academic Degree in Control Engineering in 1990 from the Academy ofSciences ofthe USSR, and Associate Professor by Teaching (Docent) Academic Degree in 1998 from the Belarusian State University of Informatics and Radio Electronics. During his career in the country he progressed from an engineer position to ChiefofJoint (Industry-Academy) Test Laboratory ofa big manufacturing enterprise (around 12 000 staD), and Head ofDepartment posts working for academy and industry. Besides he worked also as academic staD for several institutions of higher learning.

His research interests include design and test ofdigital circuits and systems, testability, fault-tolerance, digital signal processing and generation, experimental research automation, digital electronics. The list ofhis publications includes 6 books, more than 60 papers and 25 patents.

Dr. S Demidenko is Senior Member ofIEEE, Fellow and International Council Representative of IEE, and UK Chartered Engineer. Since 1998 he is a Vice-Chair ofAsia-Paci2c activities ofthe Test Technology Technical Council under the IEEE Computer Society. He was serving on the program and organising committees ofmany international conferences and workshops.

He is currently on the editorial board ofthe “JETTA: Journal ofElectronic Testing: Theory and Applications”.

Vincenzo Piuri obtained the Ph.D. in Computer Engineering in 1989, at Politecnico di Milano, Italy. From 1992 to September 2000, he was Associate Professor in Operating Systems at Politecnico di Milano. Since October 2000 he is Full Professor in Computer Engineering at the University ofMilano, Italy. He was Visiting Professor at the University ofTexas at Austin during the summers from 1993 to 1999.

His research interests include distributed and parallel computing systems, computer arithmetic, application-speci2c processing architectures, digital signal processing architectures, fault tolerance, neural network architectures, theory and industrial applications ofneural techniques for identi-2cation, prediction, control, signal and image processing. Original results have been published in more than 150 papers in book chapters, international journals, and proceedings ofinternational conferences.

He is Fellow Member ofthe IEEE and member ofACM, INNS, AEI. In the IEEE Test Technology Technical Council, he is Chair ofthe Technical Committee on Defect and Fault Tolerance. In the IEEE Instrumentation and Measurement Society, he is member ofthe Administrative Committee (1999–2001) and founding Co-Chair both of the Technical Committee on Emergent Technologies and the Technical Committee on Intelligent Measurement Systems. He is member ofthe IMACS Technical Committee on Neural Networks.

He is Associate Editor both ofthe IEEE Transactions on Instrumentation and Measurement (since 1998) and the IEEE Transactions on Neural Networks (since 2001). He was Program Co-Chair of IMTC=99 and IJCNN2000. He will be General Co-Chair ofVIMS2002 and IMTC=2004.