Fault Tolerance of Artificial Neural Networks: an Open Discussion for a Global Model

Fernando Morgado Dias and Ana Antunes

***Abstract*—** It is commonly assumed that neural networks have a built in fault tolerance property mainly due to their parallel structures. The international community of Neural Networks discussed these properties only until 1994 and afterwards the subject has been mostly ignored. Recently the subject was again brought to discussion due to the possibility of using neural networks in nano-electronic systems where fault tolerance and graceful degradation properties would be very important. In spite of these two periods of work there is still need for a large discussion around the fault model for artificial neural networks that should be used. One of the most used models is based on the stuck at model but applied to the weights. This model does not cover all possible faults and a more general model should be found. The present paper proposes a model for the faults in hardware implementations of feedforward neural networks that is independent of the implementation chosen and covers more faults than all the models proposed before in the literature.

***Keywords*—** Feedforward Neural Networks, Hardware Implementation, Fault Tolerance, Fault Model, Fault Coverage, Graceful Degradation.

I. Introduction

FAULT tolerance in Artificial Neural Networks (ANNs) has been a topic almost forgotten in the last decade. Only a few papers concerning this topic have been published after 1994: [5], [10], [1], [3], [7]. In spite of that, the intrinsic fault tolerance of Neural Networks is a very important characteristic. As stated in [9]: "The characteristic of a graceful performance degradation without additional redundancy is specially interesting for applications such as long-term, unmanned space missions, where component failures have to be expected but no repair or maintenance can be provided". Recently the utility of ANNs' built in fault tolerance was also pointed out within nano-electronic systems [5]. Beyond these areas, situations where hardware acts inside the human body can make use of both fault tolerance and graceful degradation. It is suitable that in the presence of a malfunction such hardware still performs, at least, part of its functions instead of showing complete failure.

To study fault tolerance and graceful degradation in a general way a global model for the faults is needed. Several authors have addressed this question and proposed solutions but, as will be shown here, the solutions proposed have all left uncovered some important situations.

This paper proposes the fault model that the authors believe to be more accurate and general (regardless of the implementation type used in the hardware).

The main objective of the paper is to enable the discussion of the model proposed which will be the base for future work.

The remainder of this paper is organized as follows: section 2 details the types of implementations available for ANNs, section 3 analyses the location of the possible faults in a neural network, section 3 reviews the previous work done in this field and analyses its limitations, section 4 introduces the model proposed in the present work, section 5 presents the conclusions and section 6 reports the author's perspective for future work.

Manuscript received March 5, 2008 Revised version received June 30, 2008.

F. Morgado Dias is with the Department of Mathematics and Engineering, University of Madeira, 9000-390 Funchal, Portugal (+351-291-705307; fax: +351-291-705307; e-mail: morgado@ uma.pt).

A. Antunes, is with Escola Superior de Tecnologia de Setúbal of the Instituto Politécnico de Setúbal, Campus do IPS, Estefanilha, 2910-761 Setúbal, Portugal (e-mail: aantunes@est.ips.pt).

II. Types of Implementations

This section shows the reader the types of implementations that can be found either in commercial hardware or academic implementations of ANNs. We start with a classification of the hardware solutions as proposed in [16]. This classification is represented in figure 1.

In this paper the global hardware solution is called Neurocomputers, which can be of two types: Standard Chips or Neurochips. The Standard Chips can be classified either as Sequential and Accelerator boards or Multi-Processor solutions.

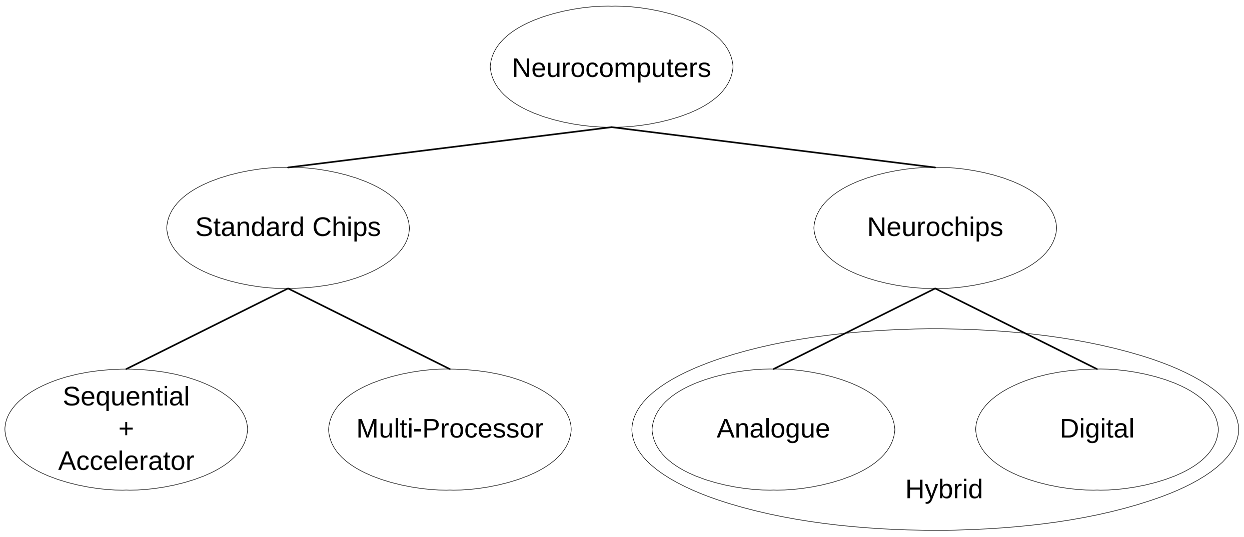

Figure 1- Neural networks' hardware categories

The Neurochips, which are constituted by Application Specific Integrated Circuits (ASICs), can be based in different kinds of circuits: Analogue, Digital or Hybrid.

The Standard Chips are digital implementations and for the present work the solution chosen for the circuit (analogue, digital or hybrid) has an important impact in analyzing the effect that the faults can have in the ANNs output.

A. Analogue Implementations

In general, analogue implementations have the advantage of obtaining high speed and density while having several technological drawbacks. These difficulties are mainly related to: store the weights, stability with temperature variations and obtaining linear multipliers over a wide operating range. Naturally these problems may result in a loss of precision when compared to other types of implementation [17], [18].

B. Digital Implementations

The digital implementations, in general, have the following advantages: weight storage in memory, easy to integrate with other applications, less difficult to implement learning algorithms and exact within the number of bits of the operands and accumulators. In a simple way, when compared with the analogue solutions, the digital ones can be labeled as more precise [18].

Most digital implementations are slower then their analogue counterparts (due to the need of conversion of the analogue inputs and to the operation of digital multipliers) and present also limitations in the implementation the activation functions which are, most of the times, simplified.

Digital implementations are usually divided into four classes: slice architectures, single instruction multiple data (SIMD), systolic array devices [19], [20], [21], [22], [23], [24] and Radial Basis Functions (RBFs), although some other classes are sometimes assumed like multiprocessor chips or others that result from the technology used (Programmable Logic Devices –PLD- and Field Programmable Gate Array - FPGA) or the similarity with other type of hardware (For example Digital Signal Processors) [18].

C. Hybrid Implementations

The hybrid implementations appear as a compromise between digital and analogue, in an attempt to obtain the best of these implementations.

Typically the external input/outputs are digital to facilitate integration into digital systems, while internally some or all the processing is analogue [20].

D. General Hardware Characteristics

There are several hardware characteristics that are important to analyze the way a fault can affect the functioning of the remaining hardware:

- The kind of storage used for internal values, inputs and weights
- The precision used for these values in the digital implementations
- The way neurons, accumulator, multipliers and activations functions are implemented

The behavior of the different implementations must be taken into account when developing a global model for fault tolerance in ANNs.

E. Examples of implementations

This subsection supplies the reader with examples of commercial implementations taken from [17] and [18].

Table 1 – Examples of commercial hardware for Artificial Neural
.

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Analogue						
Intel ETANN	Feedforward, ML	No	6bx6b	64	10280	2 GCPS
Synaptics Silicon Retina	Neuromorphic	No	N.A.	48x48	Resistive net	N.A.
Digital						
Micro Devices MD-1220	Feedforward, ML	No	1bx16b	8	8	1.9MCPS
NeuraLogix NLX-420	FeedForward, ML	No	1-16b	16	Off chip	300CPS
Philips Lneuro-1	Feedforward, ML	No	1-16b	16 PE	64	26MCPS
Philips Lneuro-2.3	N.A.	No	16-32b	12 PE	N.A.	720MCPS
Inova N64000	GP, SIMD, Int	Program	1-16b	64 PE	256k	870MCPS 220MCUPS
Hecht-Nielson HNC 100-NAP	GP,SIMD,FP	Program	32b	4 PE	512k Off chip	250MCPS 64MCUPS
Hitachi WSI	Wafer, SIMD	Hopfield	9bx8b	576	32k	138MCPS
Hitachi WSI	Wafer, SIMD	BP	9bx8b	144	N.A.	300MCUPS
Neuricam Nc3001 Totem	Feedforward, ML, SIMD	No	32b	1-32	32k	1GCPS
Neuricam Nc3003 Totem	Feedforward, ML, SIMD	No	32b	1-32	64k	750MCPS
RC Module NM6403	Feedforward, ML	Program	1-64bx1-64b	1-64	1-64	1200MCPS
Siemens MA-16	Matrix ops	No	16b	16 PE	16x16	400MCPS
Nestor/Intel NI1000	RBF	RCE, PNN, program	5b	1 PE	256x1024	40kpat/s
IBM ZISC036	RBF	ROI	8b	36	64x36	250kpat/s
Silicon Recognition ZISC 78	RBF	KNN, L1, LSUP	N.A.	78	N.A.	1Mpat/s
SAND/1	Feedforward, ML, RBF, Kohonen	No	40b	4 PE	Off-Chip	200MCPS
MCE MT19003	Feedforward, ML	No	13b	8	Off chip	32MCPS
Hybrid						
AT&T ANNA	Feedforward, ML	No	3bx6b	16-256	4096	2.1GCPS
Bellcore CLNN-32	FCR	Boltzmann	6bx5b	32	992	100MCPS 100MCUPS
Mesa Research Neuralclassifier	Feedforward, ML	No	6bx5b	6	426	21GCPS
Ricoh RN-200	Feedforward, ML	BP	N.A.	16	256	3.0GCPS

These examples are resumed in table 1, where the speed is measured as Connection Per Second (CPS) [20] which is defined as the number of multiply and accumulate operations per second during the recall or execution phase. An equivalent measure exists for the learning phase: Connection Update Per Second (CUPS) and rates the number of weight changes per second.

The diversity of implementations that was illustrated in this section makes the task of discussing a model for the faults in hardware for ANNs a very difficult one. Nevertheless the following discussion tries to focus on general model instead of considering a specific implementation.

III. Location of the Possible Faults in a Neural Network

From a functional point of view and independently of the hardware implementation options, an ANN can have the following faults:

1. Fault in a connection/weight or multiplier.

2. Fault in an input.

3. Fault in a multiplier, adder or accumulator.

4. Fault in the activation function.

The fault definition presented here is quite similar to the one proposed in [8]. Some of these faults can mask each other in the sense that it can be undistinguishable which fault occurred. This is the case for faults of type 3 and 4, which produce the same type of impact in the output and can be considered as the same situation. While other faults occur in parts of the network which have connections in parallel and the impact in the output must be analyzed, in this part of the circuit the connection is serial and the effect is similar.

Faults of type 1 and 3 both include the multiplier. This duplication was considered because the existence of a multiplier associated with each connection falls into fault type 1 and the existence of a single multiplier before the activation function falls in to fault type 3.

IV. Previous Work Regarding FeedForward Neural Networks and Limitations

As stated before, this work refers to feedforward ANNs. Figure 2 shows a very simple ANN that does not fall into this category since it has a feedback connection from the output to one of the inputs, while figure 4 has an example of a feedforward network of the Multi Input Multi Output (MIMO) type.

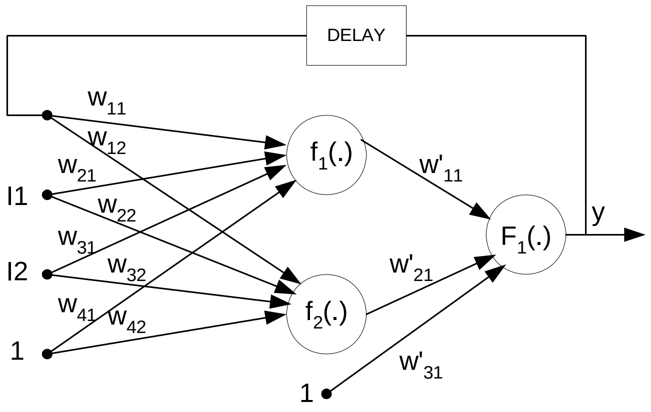

Fig. 2. Artificial Neural Network with feedback.

Most of the previous work was done regarding ANNs with binary output. The motivation for this is easy to understand since fault tolerance is much more likely to occur in this type of networks because the fault's impact in the output might be covered by the fact that the output is binary. In spite of that the present work looks for a general model that can be applied to every ANN without feedback.

In [4] the fault model used is based on modifications of the weight value of the form W(1+α) where α is allowed to take values in the interval]-1,1[. The reason to use this model is clear: it is less restrictive than most of the other solutions but it does not cover most of the possible situations. The weight itself is therefore allowed to change from 0 to 2W. Just to show an example of the limitations of this model, if a digital implementation is considered, a single fault affecting the sign bit would not be covered by this model.

In [8], although the analysis of the fault possibilities is very similar to the one proposed in section II, after a mathematical evaluation the conclusion reached is that from the behavioral point of view all the faults can be summarized as an error in the output of the neuron. This analysis is not correct since it does not hold for an error in the input because this error would affect not one but several neurons.

The paper [10] considers two types of faults: in the links or connections and in the hidden units. For the first case the faults are modeled as stuck at +MAX, 0 or –MAX, where MAX is the largest weight value in the network. For the second situation the faults are stuck at +Output_Max or -Output_Max. This model is more complete than the previous ones but the choice made for the hidden units is related to the activation functions used and is not general. The model also does not cover a fault in the inputs.

In [7] the fault model used consists of stuck at +W, -W and 0, where W is the maximum of {maximum magnitude among all the parameter values developed during learning; value needed to drive a unit into saturation}. For the bias, only +W and –W faults are considered. This model covers most of the situations but again it does not cover a fault in the input.

Several other authors used a simpler model with only a stuck at 0 possibility as pointed out in [7]. This model only covers the possibility of a missing link and ignores all other types of faults. Because of this, these models are not considered as a valid option. Their reference is not included here for simplicity's sake.

V. Fault Model Proposed

For easier perception of the following discussion figure 3 shows a representation of a neuron and figure 4 has an ANN of the MIMO type.

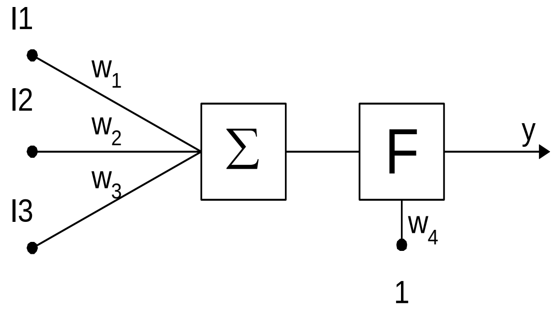

Fig. 3. Neuron structure.

Starting with a simple model similar to the one used in [9] of stuck at $+W_{MAX}$, $-W_{MAX}$ or 0 for all connections it is possible to verify what else is necessary to model all fault possibilities. This analysis will be done considering a Multi Input Multi Output (MIMO) ANN with one single fault at a time and for the sake of simplicity.

Looking at the list of possible faults in section II it is easy to verify that this model covers faults of type 1.

Faults of type 2, in one of the inputs would affect more than a single connection. If input 2 is taken as an example, it will affect the links associated with the weights w_{21} and w_{22}. This would be better modeled considering the range of the affected input and analyzing the impact of stuck at faults at $+I_{MAXN}$, $-I_{MAXN}$ and 0, where I_{MAXN} is the maximum value that the input N can assume with practical effect (for the hardware implementations there is always a limit in the maximum input due to the DACs, ADCs and power supply limits). Here each input can have a different range and it is important for the evaluation of the fault impact that the different ranges are considered.

Faults of type 3 and 4 will be analyzed together because of the similar effects they produce. The proposal here is to model these faults as faults in the weights connecting the outputs of the corresponding neurons to the next layer. This does not hold when dealing with the output neuron and contains also an approximation since the output of the neuron does not have the same range of the weights. In the first situation none of the models presented in the previous section covers an error at the output neuron and such an error is so critical that the output will have no relation with the input in most of the situations. For the second situation if the hidden layer's activation function has a limited output (which is true for a large part of the applications) the fault considered does cover appropriately the situation. Otherwise the fault will be under evaluated.

The global model to be used consists of faults of type stuck at $+W_{MAX}$, $-W_{MAX}$ or 0 for all the connections and faults of type stuck at $+I_{MAXN}$, $-I_{MAXN}$ or 0 for all the inputs, where W_{MAX} is the maximum value that the weight can assume with practical effect and I_{MAXN} is the maximum value that the input N can assume with practical effect. This model must be used in the following way:

i) Test the effect of each individual fault by setting the weight associated with each connection to $+W_{MAX}$, $-W_{MAX}$ and 0.

ii) For each input test the effect of a fault by setting the input to $+I_{MAXN}$, $-I_{MAXN}$ and 0.

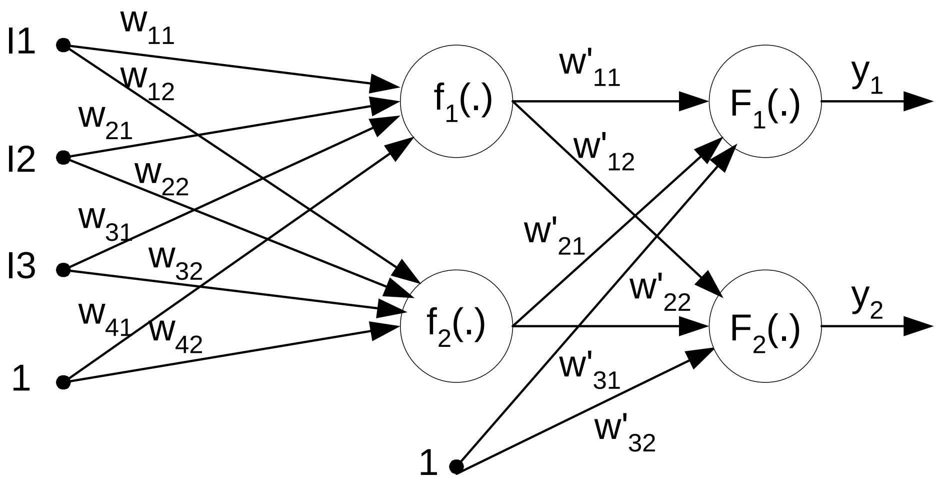

Fig. 4. Representation of a simple feedforward NN of MIMO type.

iii) For each neuron in the hidden layer(s) test the effect of a fault by setting all the weights deriving from the output of this neuron to $+W_{MAX}$, $-W_{MAX}$ and 0. This test is not needed in the case of MISO networks of only one hidden layer since then only one weight connects the output of each neuron.

This model covers more fault situations than all the models present in the literature. However the model does not cover all possible situations. One type of fault that is not covered by the model introduced here is the possibility that a connection is displaced from neuron i to neuron j. Also a fault at the output of a neuron in the output layer is not covered and faults at the output of neurons in the hidden layer with unbounded outputs may be under evaluated. Nevertheless, since the faults are evaluated at extreme situations (worst case) the model can be considered as covering with a reasonable accuracy all possible single faults.

The possibility of setting a weight to 0 to model a fault is included here more because of its physical significance (corresponds to a missing link) than because of its effect since setting the weights to $+W_{MAX}$ or $-W_{MAX}$ has a stronger impact on the output.

VI. Conclusion

The strict definition of fault tolerance does not apply to most ANNs since they do not give error free results at any time, merely an approximation [2] and appropriate models are required for accessing the effective fault tolerance of a network.

In the definition of the fault model it might be acceptable to test only the most common parts or the ones which will be responsible for the largest area [2] in silicium. Applying this principle to ANNs, due to the large number of connections, would point to testing only the weights as an acceptable solution for a large number of applications. In the present work the choice was to try to cover all possible situations and the analysis of each possible fault led to a different contribution to the final model.

The model presented here covers more situations than the models previously discussed in the literature. These models all failed to evaluate the effect of a fault in the inputs and under evaluate the faults in the neurons in the hidden layer. Although the present model has some limitations (considers only single faults, faults in neurons in the hidden layer with unbounded outputs may be under evaluated, faults at the output neuron are not considered) it is more accurate than the previous ones.

The existence of global models for faults is of extreme importance to permit evaluation and comparison of different networks, training algorithms and special modification algorithms regarding fault tolerance of ANNs.

One limitation of the present work, which also occurs in the previous reported papers, is that only setting the weights to zero, maximum value and minimum value is considered. Since the ANN main output does not need to be a monotonic or a linear function of the weights and inputs, other combinations may occur which produce lower fault tolerance than the maximum one reported with this model.

VII. Future Work

As future work, the authors intend to propose a methodology to improve fault tolerance of ANNs.

It is possible to study the solutions to improve the fault tolerance capabilities. At least the following possibilities exist:

i) Change the training algorithms.

ii) Dynamic reconfiguration processes to be applied after a fault occurs.

iii) Modify the ANN after the training stage without changing its architecture.

iv) Modify the ANN after the training stage changing its architecture.

The possibility of changing the training algorithms to obtain a better fault tolerance was also the object of study of some researchers: [3], [14], [6].

The remaining solutions deal with the possibility of changing the ANN after the training stage. These can be implemented with or without changing the architecture of the network.

The dynamic reconfiguration option obliges the use of on-line training algorithms and also on-line detection of faults which is not at all a straightforward task to implement. This on-line detection of faults might be obtained through the use of an enlarged Boundary Scan scheme [15] and test information with extensive coverage to allow the identification of the fault and the reorganization of the ANN without the damaged elements.

The solution to improve fault tolerance after the training stage without changing the network's architecture is theoretically possible (at least in some nodes, but depending on the ANN structure), though difficult to implement since the changes must be such that the network's answers are the same before and after the change for every possible input.

After elaborating a strategy for improving fault tolerance of ANNs, this strategy can then be used as a base to build an automatic tool to improve fault tolerance of ANNs.

VIII. References:

[1] B. S. Arad and A. El-Amawy, "On Fault Tolerance Training of Feedforward Neural Networks", Neural Networks, Vol. 10/3, pp.539-553, 1997.

[2] G. Bolt, "Investigating the Fault Tolerance in Artificial Neural Networks", Technical Report YCS 154 from the University of York, available at the internet, 1991.

[3] S. Cavalieri and O. Mirabella, "A novel learning algorithm which improves the partial fault tolerance of multiplayer neural networks", Neural Networks, Vol.12, pp. 91-106, 1999.

[4] C. Chiu, K. Mehrotra, C. Mohan and S. Ranka, "Robustness of Feedforward Neural Networks", 2nd IEEE International Conf. on Neural Networks, Vol. 2, pp. 783-788, 1993.

[5] R. Eickhoff and U. Rückert, "Tolerance of Radial Basis Functions Against Stuck-at-Faults", International Conference of Artificial Neural Networks, pp. 1003-1008, Poland, 2005.

[6] H. Elsimary, S. Mashali and S. Shaheen, "Generalization Ability of Fault Tolerant Feed Forward Neural Networks", International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 30-34, 1995.

[7] D. S. Phatak, "Complete and Partial Fault Tolerance of Feedforward Neural Nets", IEEE Transactions on Neural Networks, Vol. 6/2, pp. 446-456, 1995.

[8] V. Piuri, M. Sami, R. Stefanelli, "Fault Tolerance in Neural Networks: Theoretical Analysis and Simulation Results", 'Advanced Computer Technology, Reliable Systems and Applications', 5th Annual European Computer Conference, Bologna, Italy, 1991.

[9] E. B. Tchernev and D. S. Phatak, "Investigating the fault tolerance of neural networks", NCA, 2005.

[10] P.W. Protzel, D. L. Palumbo and M. K, Arras, "Performance and fault tolerance of neural networks for optimization", IEEE transactions on Neural Networks, vol.4, 1993.

[11] F. Morgado Dias, A. Antunes and A. Mota, "A new hybrid direct/specialized approach for generating inverse neural models", 6th WSEAS Int. Conf. on Algorithms, Scientific Computing, Modelling and Computation (ASCOMS'04), Cancun, México, 2004.

[12] A. Antunes, F. Morgado Dias, A. Mota, "Influence of the sampling period in the performance of a real-time distributed system under jitter conditions", 6th WSEAS Int. Conf. on Telecommunications and Informatics (TELE-INFO'04), Cancun, 2004.

[13] J. Vieira, F. Morgado Dias, A. Mota,"Neuro-Fuzzy Systems: A Survey", 5th WSEAS NNA International Conference on Neural Networks and Applications, Udine, Italia, 2004.

[14] H. Elsimary, S.Mashali, and S.Shaheen. Performance evaluation of a novel fault tolerance training algorithm. proceedings of the World congress on computational intelligence, 2:856—860, 1994.

[15] K. Parker, "The Boundary Scan Handbook", 1992.

[16] Heemskerk, J. N. H., "Overview of Neural Hardware. Neurocomputers for Brain-Style Processing. Design, Implementation and Application", PhD Thesis, Unit of Experimental and Theoretical Psychology, Leiden University, the Netherlands, Draft version available from ftp.mrc-apu.cam.ac.uk/pub/nn, 1995.

[17] F. Morgado Dias, A. Antunes and A. Mota, "Artificial Neural Networks: a Review of Commercial Hardware", Engineering Applications of Artificial Intelligence, IFAC, Vol. 17/8, pp. 945-952, 2004.

[18] F. Morgado Dias, A. Antunes and A. Mota, "Commercial Hardware for Artificial Neural Networks: a Survey", SICICA - 5th IFAC International Symposium on Inteligent Components and Instruments for Control Applications, Aveiro, Portugal, 2003.

[19] D. Baratta, G. M. Bo, D. D. Caviglia, f. Diotalevi and M. Valle, "Microelectronic Implementation of Artificial Neural Network", Invited paper in the proceedings of the 5th Electronics Devices and Systems International Conference, Brno, Czech Republic, June 1998.

[20] C. Lindsey, and T. Lindblad, "Review of Hardware Neural Networks: A User's Perspective", Proceeding of 3rd Workshop on Neural Networks: From Biology to High Energy Physics, Isola d'Elba, Italy, Sept. 26-30, 1994.

[21] C. Lindsey, B. Denby and T. Lindblad, "Neural Networks in HEP", available in the internet at http://www1.cern.ch/NeuralNets/nnwInHepHard.html.

[22] C. Lindsey, "Neural Networks in Hardware: Architectures, Products and Applications", available in the internet at http://www.particle.kth.se/~lindsey/HardwareNNWCourse/home.html

[23] C. Lindsey, B. Denby and T. Lindblad, " FutureAI.com – Artificial Intelligence – Neural Network Hardware", available in the internet at fttp://www.futureai.com/tutorials/hardware.php

[24] C. Lindsey, T. Lindblad, "Survey of Neural Network Hardware", Proc. SPIE Vol. 2492, p. 1194-1205, Applications and Science of Artificial Neural Networks, Steven K. Rogers; Dennis W. Ruck; Eds. , March 1995.

[25] Neural Networks: Commercial Hardware Tools", available in the internet at http://www..emsl.pnl.gov:2080/proj/neuron/neural/ systems/commercial.html

[26] NEuroNet Research Committee, "A European Strategy for Neural Networks", available in the internet at http://www.kcl.ac.uk/neuronet/about/roadmap/hardware.html

[27] Digital Neural Networks in VLSI", Module Research Center, available in the internet at http://www.module.ru

[28] "FAQ on Neural Networks part 7 – Hardware NNW", available in the internet at ftp//:ftp.sas.com/pub/neural/FAQ.html

[29] P. Ienne and G. Kuhn, "Digital Systems for Neural Networks", P. Papamichalis and R. Kerwin, editors, Digital Signal Processing Technology, volume CR57, of Critical Reviews Series, pag. 314-345, SPIE Optical Engineering, Orlando, Fla. 1995.

[30] P. Jedrasik, "Hardware implementation for real time RBF-based proximity effects neurocorrector", chapter 8 of "Proximity Effects Correction in Electron Beam Nanolithography, University of Anrwerpen, Departement Natuurkunde, Antwerpen, 1998, available in the internet at http://143.129.203.3/visielab/theses/jedrasik/chap80.pdf

[31] E. van Keulan, S.Colak, H. Withagen and H. Hegt, "Neural Networks Hardware Performance Criteria", Proceedings of the IEEE Conference on Neural Networks, III, 1885-1888, 1994.

[32] Y. Liao, Y., "Neural Networks in Hardware: A Survey", available in the internet at http://wwwcsif.cs.ucdavis.edu/~liaoy/research/NNhardware.pdf

[33] T. Schoenauer, A. Jahnke, U. Roth and H. Klar, "Digital Neurohardware: Principles and Perspectives", Neuronal Networks in Applications, Magdeburg, 1998.

[34] R. Schüffny, A. Graupner and J. Schreiter, "Hardware for Neural Networks", 4th International Workshop Neural Networks in Applications, Magdeburg, Germany, 1999.

[35] M. Taveniku, M., and A. Linde, " A Reconfigurable SIMD Computer for Artificial Neural Networks", Technical Report nº189L, Chalmers University of Technology.

Fernando Morgado Dias received his engineering degree in Electronics and Telecomunications from the University of Aveiro, Portugal (1994), his *Diplôme D'Études Approfondies* in Microelectronics from the University Joseph Fourier in Grenoble, France (1995) and his PhD in Electrical Engineering from the University of Aveiro, Portugal (2005). His main research field is artificial neural

He is a former teacher from the Superior School of Technology of Setubal, Portugal and is currently Assistant professor at the University of Madeira, Portugal. He has published over 40 scientific papers in the fields of control and neural networks.
Prof. Dias has received two awards, the best paper in session CCS-07 Robust Controllers I, in the 28th Annual Conference of the IEEE Industrial Electronics Society conference, with the paper "Additive Internal Model Control: an Application with Neural Models in a Kiln", held in Sevilha, Spain, 2002 and second place in the best design student project category with the integrated circuit "AMBROSIO: Advanced Microchip for Broadband System Improvement and Optimization", in the 5th Eurochip Workshop, Dresden, Germany 1994.

Ana Antunes received her engineering degree in Electronics and Telecomunications from the University of Aveiro, Portugal (1994), her *Diplôme D'Études Approfondies* in Microelectronics from the University Joseph Fourier in Grenoble, France (1995) and her PhD degree in Electrical Engineering at the University of Aveiro, Portugal (2008). Her main research field is distributed control systems.

She currently teaches at the Superior School of Technology of Setúbal, Portugal and has published over 30 scientific papers in the fields of distributed control and neural networks.
Prof. Antunes has received three awards: best paper in the conference ANIPLA 2006- Methodologies for Emerging Technologies in Automation with the paper "Dynamic Rate Adaptation in Distributed Computer Control Systems", held in Roma, Italy; best paper in session CCS-07 Robust Controllers I, in the 28th Annual Conference of the IEEE Industrial Electronics Society conference, with the paper "Additive Internal Model Control: an Application with Neural Models in a Kiln", held in Sevilha, Spain, 2002 and second place in the best design student project category with the integrated circuit "AMBROSIO: Advanced Microchip for Broadband System Improvement and Optimization", in the 5th Eurochip Workshop, Dresden, Germany 1994.