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.

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.
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>
<thead>
<tr>
<th>Name</th>
<th>Architecture</th>
<th>Learn</th>
<th>Precision</th>
<th>Neurons</th>
<th>Synapses</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analogue</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel ETANN</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>6bx6b</td>
<td>64</td>
<td>10280</td>
<td>2 GCPS</td>
</tr>
<tr>
<td>Synaptics Silicon Retina</td>
<td>Neuromorphic</td>
<td>No</td>
<td>N.A.</td>
<td>48x48</td>
<td>Resistive net</td>
<td>N.A.</td>
</tr>
<tr>
<td><strong>Digital</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micro Devices MD-1220</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>1bx16b</td>
<td>8</td>
<td>8</td>
<td>1.9MCPS</td>
</tr>
<tr>
<td>NeuraLogix NLX-420</td>
<td>FeedForward, ML</td>
<td>No</td>
<td>1-16b</td>
<td>16</td>
<td>Off chip</td>
<td>300CPS</td>
</tr>
<tr>
<td>Philips Lineuro-1</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>1-16b</td>
<td>16 PE</td>
<td>64</td>
<td>26MCPS</td>
</tr>
<tr>
<td>Philips Lineuro-2.3</td>
<td>N.A.</td>
<td>No</td>
<td>16-32b</td>
<td>12 PE</td>
<td>N.A.</td>
<td>720MCPS</td>
</tr>
<tr>
<td>Inova N64000</td>
<td>GP, SIMD, Int Program</td>
<td>1-16b</td>
<td>64 PE</td>
<td>256k</td>
<td>870MCPS</td>
<td>220MCUPS</td>
</tr>
<tr>
<td>Hecht-Nielson HNC 100-NAP</td>
<td>GP, SIMD, FP Program</td>
<td>32b</td>
<td>4 PE</td>
<td>512k</td>
<td>Off chip</td>
<td>250MCPS</td>
</tr>
<tr>
<td>Hitachi WSI</td>
<td>Wafer, SIMD Hopfield</td>
<td>9bx8b</td>
<td>576</td>
<td>32k</td>
<td>138MCPS</td>
<td></td>
</tr>
<tr>
<td>Hitachi WSI</td>
<td>Wafer, SIMD BP</td>
<td>9bx8b</td>
<td>144</td>
<td>N.A.</td>
<td>300MCUPS</td>
<td></td>
</tr>
<tr>
<td>Neuricam Nc3001 Totem</td>
<td>Feedforward, ML, SIMD</td>
<td>No</td>
<td>32b</td>
<td>1-32</td>
<td>32k</td>
<td>1GCPS</td>
</tr>
<tr>
<td>Neuricam Nc3003 Totem</td>
<td>Feedforward, ML, SIMD</td>
<td>No</td>
<td>32b</td>
<td>1-32</td>
<td>64k</td>
<td>750MCPS</td>
</tr>
<tr>
<td>RC Module NM6403</td>
<td>Feedforward, ML Program</td>
<td>1-64bx1-64b</td>
<td>1-64</td>
<td>1-64</td>
<td>1200MCPS</td>
<td></td>
</tr>
<tr>
<td>Siemens MA-16</td>
<td>Matrix ops No</td>
<td>16b</td>
<td>16 PE</td>
<td>16x16</td>
<td>400MCPS</td>
<td></td>
</tr>
<tr>
<td>Nestor/Intel N11000</td>
<td>RBF RCE, PNN, program</td>
<td>5b</td>
<td>1 PE</td>
<td>256x1024</td>
<td>40kpat/s</td>
<td></td>
</tr>
<tr>
<td>IBM ZISC036</td>
<td>RBF ROI</td>
<td>8b</td>
<td>36</td>
<td>64x36</td>
<td>250kpat/s</td>
<td></td>
</tr>
<tr>
<td>Silicon Recognition ZISC 78</td>
<td>RBF KNN, L1, LSUP</td>
<td>N.A.</td>
<td>78</td>
<td>N.A.</td>
<td>1Mpat/s</td>
<td></td>
</tr>
<tr>
<td>SAND/1</td>
<td>Feedforward, ML, RBF, Kohonen</td>
<td>No</td>
<td>40b</td>
<td>4 PE</td>
<td>Off-Chip</td>
<td>200MCPS</td>
</tr>
<tr>
<td>MCE MT19003</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>13b</td>
<td>8</td>
<td>Off chip</td>
<td>32MCPS</td>
</tr>
<tr>
<td><strong>Hybrid</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT&amp;T ANNA</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>3bx6b</td>
<td>16-256</td>
<td>4096</td>
<td>2.1GCPS</td>
</tr>
<tr>
<td>Bellcore CLNN-32</td>
<td>FCR Boltzmann</td>
<td>6bx5b</td>
<td>32</td>
<td>992</td>
<td>100MCPS</td>
<td>100MCUPS</td>
</tr>
<tr>
<td>Mesa Research Neuralclassifi er</td>
<td>Feedforward, ML</td>
<td>No</td>
<td>6bx5b</td>
<td>6</td>
<td>426</td>
<td>21GCPS</td>
</tr>
<tr>
<td>Ricoh RN-200</td>
<td>Feedforward, ML BP</td>
<td>N.A.</td>
<td>16</td>
<td>256</td>
<td>3.0GCPS</td>
<td></td>
</tr>
</tbody>
</table>
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

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 \( W(1+\alpha) \) where \( \alpha \) 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 \( +\text{MAX}, 0 \) or \( -\text{MAX} \), where \( \text{MAX} \) is the largest weight value in the network. For the second situation the faults are stuck at \(+\text{Output Max} \) or \(-\text{Output Max}\). 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.

For testing purposes, a simple model similar to the one used in [9] of stuck at $+W_{\text{MAX}}$, $-W_{\text{MAX}}$ or 0 for all connections 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 $+I_{\text{MAXN}}$, $-I_{\text{MAXN}}$ and 0, where $I_{\text{MAXN}}$ 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 sections 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 $+W_{\text{MAX}}$, $-W_{\text{MAX}}$ or 0 for all the connections and faults of type stuck at $+I_{\text{MAXN}}$, $-I_{\text{MAXN}}$ or 0 for all the inputs, where $W_{\text{MAX}}$ is the maximum value that the weight can assume with practical effect and $I_{\text{MAXN}}$ 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 $+W_{\text{MAX}}$, $-W_{\text{MAX}}$ and 0.

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


Fernando Morgado Dias received his engineering degree in Electronics and Telecommunications 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 networks.

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 Telecommunications 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.