Intrinsic Fault Tolerance of Hopfield Artificial Neural Network Model for Task Scheduling Technique in SoC

Rajhans Singh[1] and Daniel Chillet[2]
[1]*Indian Institute of Technology Roorkee, Roorkee, 247667, India*
[2]*CAIRN, IRISA/INRIA, University of Rennes 1, ENSSAT, 6 rue de Kerampont, BP 80518, 22305 Lannion, France*

Keywords: Fault Tolerance of Hopfield Articifial Neural Network (HANN), Task Scheduling with HANN, Optimization Problem.

Abstract: Due to the technology evolution, one of the main problems for future System-on-Chips (SoC) concerns the difficulties to produce circuits without defaults. While designers propose new structures able to correct the faults occurring during computation, this article addresses the control part of SoCs, and focuses on task scheduling for processors embedded in SoC. Indeed, to ensure the execution of application in presence of faults on such systems, operating system services will need to be fault tolerant. This is the case for the task scheduling service, which is an optimization problem that can be solved by Artificial Neural Network. In this context, this paper explores the intrinsic fault tolerance capability of Hopfield Artificial Neural Network (HANN). Our work shows that even if some neurons are in fault, a HANN can provide valid solutions for task scheduling problem. We define the intrinsic limit of fault tolerance capability of Hopfield model and illustrate the impact of fault on the network convergence.

1 INTRODUCTION

Nowadays, classical current embedded applications require high performance systems and highly integrated solutions. The large computation needs find response through the technology evolution which enables to integrate more and more transistors in each mm^2. But this technology evolution leads to the design of circuits which can include/embed faults during the fabrication process. In this case, designers must define circuits able to continue to produce valid computations in presence of faults. These solutions must address the computation part but also the control part of such circuits. While a large number of studies have addressed the computation part of SoC (processor, memory, ...), this paper addresses the control part and proposes a specific operating system service for the tasks scheduling. The objective consists in ensuring the schedule of tasks of the application on the processor available on the SoC, even if some faults appear in the task scheduling service which is in charge of the execution control of the application.

The scheduling process can be treated as optimization problem, and numerous algorithms have been developed for this problem. Hopfield Artificial Neural Networks (HANN) and genetic algorithms are interesting tools for solving optimisation problems. One of the work in this field comes from Cardeira in (Cardeira and Mammeri, 1995). In this paper, the authors used HANN for tasks scheduling for monoprocessor (see figure 1). The main difficulty to solve optimization problems with ANN consists in finding a good representation for the problem. In the figure 1.a, the scheduling problem is represented by a matrix of neurons, where each line represents a task and each column represents a schedule cycle. For this representation, one task is added to represent the processor inactivity (fictive task) when the load is not equal to 100%. Figure 1.b represents a specific solution for the scheduling problem. For each task, the number of active neurons (black neurons) is equal to the number of cycles needed for the task (values k_i). Furthermore, for each column, the number of active neurons can not be greater than 1, which means that only one task can be active at each schedule cycle. This model can be used to support homogeneous multiprocessors by adding several fictive tasks in the matrix of neurons. In (Chillet et al., 2011) and (Chillet et al., 2010), the authors use HANN for real time scheduling on heterogeneous SoC architecture.

The capability to solve optimization problems by using ANN is completed by intrinsic capability of

Singh R. and Chillet D..
Intrinsic Fault Tolerance of Hopfield Artificial Neural Network Model for Task Scheduling Technique in SoC.
DOI: 10.5220/0005140902880293
In *Proceedings of the International Conference on Neural Computation Theory and Applications* (NCTA-2014), pages 288-293
ISBN: 978-989-758-054-3

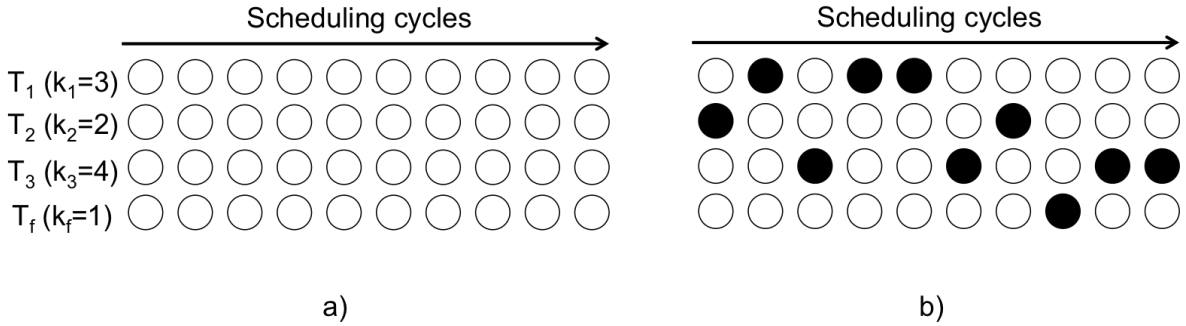

Figure 1: Task scheduling problem represented by HANN. a) Problem representation: each circle is a neuron of HANN, each row represent a task with different k value and each column represents a scheduling cycle. b) One possible solution for the optimization problem: each dark circle represent an active cycle time for the corresponding task.

ANN to support faults. This feature has been demonstrated for HANN in (Protzel et al., 1993) (Bolt, 1992), and (Tchernev et al., 2005). The work presented in (Protzel et al., 1993) demonstrates the fault tolerance capability of recurrent ANN model (such as Hopfield model) for optimization problem like the traveling Salesman Problem and the assignment problem. This paper shows that intrinsic fault tolerance of recurrent networks depend on the type of optimization problem. The authors of (Bolt, 1992) investigate the inherent fault tolerance of neural networks which arises from their unusual computational features, such as distribution of information, generalization. It also models the fault into different level like electrical, logical and functional. It also shows that Multilayer Perceptron ANN has intrinsic fault tolerance only when appropriate method of training is used for the network. Adding fault tolerance constraints during the training also help in improving the global behavior of the neural network. The paper (Tchernev et al., 2005) shows how improving the fault tolerance of a feed forward ANN for optimization problem by modifying the training method of the network In (Kamiura et al., 2004), the authors propose to define the network by taking into account of faults occurring. Single-fault injection and double-fault injection are used for learning schemes and the authors demonstrate the the fault tolerance is improved. In our context, the HANN is defined without learning scheme, and the objective is to support internal fault of the network. This objective comes from the current evolution of technology which places the designer face to the need to manage faults occurring in the circuits. The fault tolerance capability of HANN opens a new opportunity to implement them and to exploit their intrinsic characteristic. This paper addresses this topic and illustrates that task scheduling can be obtained by using ANN even if faults are present in the ANN.

The type of faults addressed in this work concerns permanent faults due to the incorrect behavior of some transistors which implement the ANN. This type of faults in ANN implementation can cause i) input signal of a neuron to be too low or too high, ii) neural connection weights to be too low or too high or iii) fault inside a neuron. Considering these faults, we claim that ANN can support a limited number of faults, this number depends on the task's characteristics. This paper formulated this limit and shows the impact of faults on the convergence process.

The remainder of the paper is organized as follows. In section 2, we present the types of faults that can occur in hardware implementation of HANN. Section 3 presents the mathematical formulation for the intrinsic limit on the number of faults which can be supported. Section 4 presents experimental results of the effect of faults on the convergence time. Section 5 concludes this paper and presents some perspectives.

2 TYPES OF FAULT IN ANN

This section analyzes the different categories of simple faults (considered in this work) which can occur in ANNs. From this analysis, we propose to limit the faults as only two types. The figure 2 summarizes these different types of faults on a simple neuron network, and the following points detail the different categories of faults and explain the impact on ANN behavior.

- Permanent fault occurring inside the neuron: in this case, the neuron state remains fixed. This type of fault will make the neuron to be in permanently active or inactive state. in the figure 2, this case is represented by the default on the neuron N_1.
- Permanent fault occurring on the input signal of the neuron (input is too low or too high): in this case, low magnitude input causes the neuron to

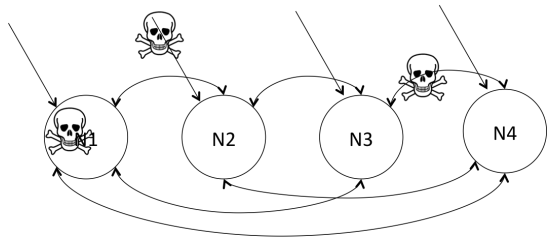

Figure 2: Representation of types of faults which can occurred in the neural network.

remain always inactive and high magnitude input causes the neuron to be always active.in the figure 2, this case is represented by the default on the input of neuron N_2.

- Permanent fault occurring in the connection between two neurons (connection weight is too high or too low): in this case, a neuron which receives a high negative connection value from one of its connections will be always placed in inactive state. Contrary, a neuron which receives high positive connection value will be always placed in active state. in the figure 2, this case is represented by the default on the connexion between neurons N_3 and N_4. For this type of fault, we consider that the the bidirectional connexion is implemented by two uni-directional connexions, and in this case the fault impacts just one neuron.

Considering these different categories, we can simplify the model of ANN fault as only two simple cases: i) always active neuron type, ii) always inactive neuron type. In the following sections, we consider only these two types of faults.

3 FAULT TOLERANCE OF ANN FOR TASKS SCHEDULING CONTEXT

Tasks scheduling is a classical optimization problem and ANN is known as an interesting tool for solving this type of problems due to its parallel computing characteristic which makes it faster than any other method. For example, Cardeira (Cardeira and Mammeri, 1995) explains how HANN can be used for the scheduling problem (see figure 1). To solve this problem, the ANN is defined by a set of neurons and by an energy function optimized during the convergence step. This convergence leads to energy minima (Hopfield, 1984; Hopfield and Tank, 1985) which represents task scheduling solution.

In HANN containing n neurons, each neuron n_i is connected to all other neurons n_j by a weight W_{ij} and receives an input energy I_i (Hopfield, 1984). The evolution of state x_i of neuron n_i and the energy function of the network is given by:

$$x_i = \begin{cases} 1, & \text{if } I_i + \sum_{j=0}^{n} x_j \cdot W_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$E = -\frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{n} W_{ij} \cdot x_i \cdot x_j - \sum_{i=0}^{n} I_i \cdot x_i \quad (2)$$

In (Cardeira and Mammeri, 1995) author shows that the solution for scheduling problem can be obtained by applying a specific $k-outof-N$ rule on the ANN. The energy function of this rule is:

$$E_{k-outof-N} = (k - \sum_{i=0}^{n} x_i)^2 \quad (3)$$

This energy function can be written in form of equation 2 to find the connection weights and the input values:

$$E_{k-outof-N} = -\frac{1}{2}\sum_{i=0}^{n}\sum_{j=0 j\neq i}^{n} (-2) \cdot x_i \cdot x_j - \sum_{i=0}^{n} (2k-1) \cdot x_i + k^2$$

The term k^2 is a constant offset and it has no influence on the energy minima. For analysing the effect of faults on ANN, let us suppose there are T number of rows and C number of columns in the ANN (T tasks and C cycles), and k_{ri} and k_{ci} are the corresponding k values for the i^{th} rows and i^{th} column. Also, suppose n_{1ri} and n_{1ci} are the number of neurons which are always active (fault) in the i^{th} row and i^{th} column. Let, x_{ri} and x_{ci} be the number of active neurons which are not in fault in the i^{th} row and i^{th} column respectively. So the energy function can be written as:

$$\begin{aligned} E &= \sum_{i=0}^{T}((n_{ri}+x_{ri})^2 - 2k_{ri}(x_{ri}+n_{ri})) \\ &\quad + \sum_{i=0}^{C}((n_{ci}+x_{ci})^2 - 2k_{ci}(x_{ci}+n_{ci})) \end{aligned} \quad (4)$$

According to the Hopfield proposition, network converges towards a point where energy function is minimum. By computing the derivative function of this energy function with respect to x_{ri} and x_{ci}, and by equating it to the zero, we can obtain the minima point corresponding to that row or column.

$$\begin{aligned} \frac{dE}{dx_{ri}} &= 2(n_{ri}+x_{ri}) - 2k_{ri} \\ x_{ri} &= (k_{ri} - n_{ri}) \end{aligned} \quad (5)$$

$$\begin{aligned} \frac{dE}{dx_{ci}} &= 2(n_{ci}+x_{ci}) - 2k_{ci} \\ x_{ci} &= (k_{ci} - n_{ci}) \end{aligned} \quad (6)$$

From the expression given in equation 5, as the value of x_{ri}, number of active states in a row which are not in fault cannot be negative, so for the network, to provide a valid solution we can say that n_{1ri} , number of always active faulted neurons, should be less than or equal to k_{ri}. Similarly for the column, we can say that n_{1ci} should be less than or equal to the k_{ci}.

For always inactive type of faults given in equation 5 and 6, making n_{1ri} and n_{1ci} to zero and taking the maximum value of x_{ri} and x_{ci} to be C_{n2ri} and T_{n2ci}, where C_{n2ri} is number of neurons in the i^{th} row which are not in always inactive type fault and T_{n2ci} is number of neurons in the i^{th} column which are not in always inactive type fault. From this we can say that the number of inactive type faults in a particular row i^{th} should be less than the $(C - k_i)$ value of that row or in case of column j^{th} it should be less than $(T - k_j)$ value. If the number of faults satisfies the above conditions, the network will provide the valid solution even if some of the neurons are in fault.

4 EXPERIMENTAL RESULTS

To illustrate the fault tolerance capability, we simulate the Hopfiel ANN for tasks scheduling with a periodic cycle of 20 and 10 number of tasks. Considering that ANN is defined by applying several rules on the same neurons of the network, the ANN can converge to local minima instead of global minima. This situation can be avoided by adding extra energy to the system, which can be easily achieved by re-initialization of all neurons. So network takes few re-initializations to converge to the valid solution. In this case, performance of network is measured in terms of numbers of initializations and iterations required to converge to valid solution.

Figures 3 and 4 show that when the number of always inactive type of faults increases in a row or column, the number of iterations and re-initializations needed for the network to provide the valid solution increases. When the number of always inactive type of faults in a particular row or column is less than the $(N - k_i)$ value then the network does provide the valid solution but the number of re-initializations increases very rapidly. This means that increasing the always inactive type of fault increases the amount of local minima in the energy function. For the neurons which are connected with zero weight value, increasing the always inactive type faults in these neurons doesn't affect the numbers of iterations and the re-initializations.

Figure 5 shows that increasing always active type of faults in ANN increases the number of iterations

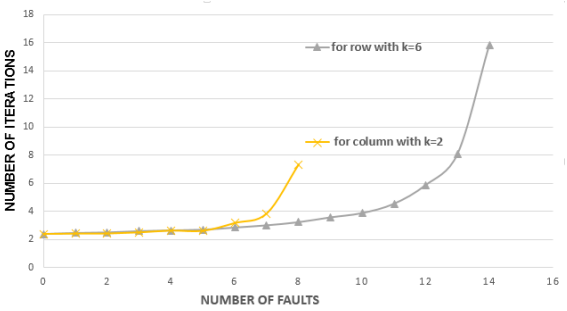

Figure 3: Effect of faults on the average numbers of iterations needed for a task scheduling problem consisting of 20 scheduling cycles (nb of columns) and 10 tasks (nb of rows). 1000 simulations are made for all points.

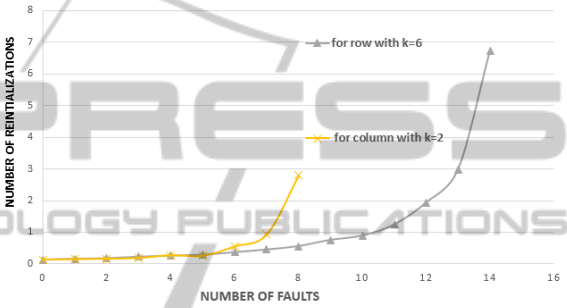

Figure 4: Effect of faults on the average numbers of reinitializations needed for a task scheduling problem consisting of 20 scheduling cycles (nb of columns) and 10 tasks (nb of rows). 1000 simulations are made for all points.

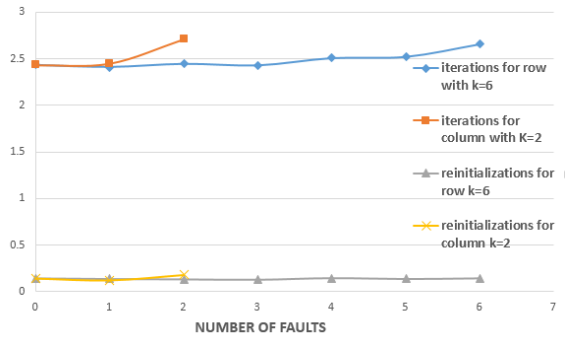

Figure 5: Numbers of re-initalizations and iterations required to provide a valid solution for 1000 simulations in a network of size 20 cycles (columns) and 10 tasks (rows).

but this increase is not significant. Network provides the solution when the number of faults in a particular row or column remains less than the k_i value of that row or column. The number of re-initializations, in this case, remains almost same. Thus, we can say that always active type of faults does not affect the amount of local minima in the energy function. From this, we have shown that ANN provides the valid solution when the number of faults does not violate the intrinsic limit of fault tolerance. Figures 7 shows that

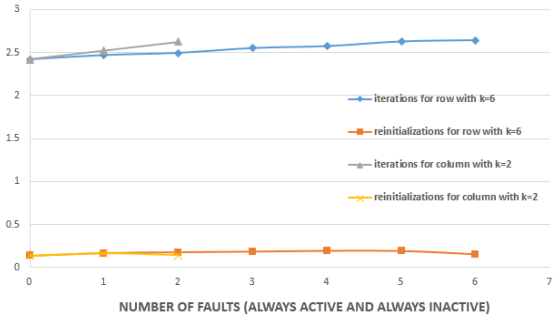

Figure 6: Numbers of re-initalizations and iterations required to provide a valid solution when same number of always active type faults and always inactive type faults added in the same row and column in a network of size 20 cycles (columns) and 10 tasks (rows).

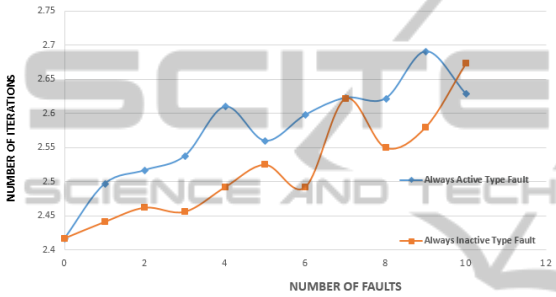

Figure 7: Numbers of iterations required to provide a valid solution when faults are added to the neurons which are not connected with each other (zero connection value) in a network of size 20 cycles (columns) and 10 tasks (rows).

when faults are distributed to neurons which are not connected with each other do not affect the number of iterations required to provide the valid solution significantly. It also shows that the network can provide valid solution with almost same number of required iterations even with large number of faults. But these faults should be distributed evenly.

Figures 6 shows that if the same number of always active and in active type faults present in the same row or column the number of iterations required to provide the solution do not changes significantly in compare with presence of only inactive type fault. This also shows that only if there is dominance of same kind of fault in a particular row of column then only the number of iterations required to provide the solution varies significantly.

5 CONCLUSIONS

In this article, we have demonstrated that intrinsic fault tolerance capability of HANN can be used in the context of tasks scheduling for future SoC which will be produced with very important technology variability and need to support faults. We have shown that even if some neurons are in fault in the ANN, the network is able to provide valid solutions. We have defined the limit of intrinsic fault tolerance of HANN in this context of tasks scheduling. We have also shown that number of iterations required to provide the solution does not increase significantly in case of always active type of faults. But in case of always inactive type of faults, the numbers of iterations and re-initializations required to provide valid solution increase very fast when the amount of faults is closer to the intrinsic limit.

This paper shows that the HANN has good fault tolerance capability for the scheduling problem. But there might be some cases when the total number of faults are quite high and it mainly consists of always inactive type of faults. In such cases, the number of iterations and re-initializations required to provide the solution is quite high. Also in some extreme cases, the number of faults can violate the intrinsic limit of fault tolerance. So, future work in this field can be detection of the intrinsic fault tolerance limit violation from the state of neuron after first or second convergence. So we can analyze wether the network will provide valid solution or not after few convergence and we can define propose solution to remove these faults from the network.

REFERENCES

Bolt, G. R. (1992). *Fault tolerance in artificial neural networks: are neural networks inherently fault tolerant?*. PhD thesis, University of York.

Cardeira, C. and Mammeri, Z. (1995). Preemptive and non-preemptive real-time scheduling based on neural networks. *Proceedings DDCS95*, pages 67–72.

Chillet, D., Eiche, A., Pillement, S., and Sentieys, O. (2011). Real-time scheduling on heterogeneous system-on-chip architectures using an optimised artificial neural network. *Journal of Systems Architecture*, 57:340–353.

Chillet, D., Pillement, S., and Sentieys, O. (2010). *Algorithm-Architecture Matching for Signal and Image Processing, Springer*, volume 73 of *Lecture Notes in Electrical Engineering*, chapter RANN: A Reconfigurable Artificial Neural Network Model for Task Scheduling on Reconfigurable System-on-Chip, pages 117–144. Springer Netherlands.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092.

Hopfield, J. J. and Tank, D. W. (1985). neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152.

Kamiura, N., Isokawa, T., and Matsui, N. (2004). On improvement in fault tolerance of hopfield neural networks. In *Test Symposium, 2004. 13th Asian*, pages 406–411.

Protzel, P. W., Palumbo, D. L., and Arras, M. K. (1993). Performance and fault-tolerance of neural networks for optimization. *Neural Networks, IEEE Transactions on*, 4(4):600–614.

Tchernev, E. B., Mulvaney, R. G., and Phatak, D. S. (2005). Investigating the fault tolerance of neural networks. *Neural computation*, 17(7):1646–1664.