Hardware Implementation of a Fault-Tolerant Hopfield Neural Network on FPGAs

Juan Antonio Clemente[a], Wassim Mansour[b], Rafic Ayoubi[c], Felipe Serrano[a], Hortensia Mecha[a], Haissam Ziade[d], Wassim El Falou[d], Raoul Velazco[b]

[a] *Computer Architecture Department, Universidad Complutense de Madrid, Spain*
[b] *TIMA Laboratory, INPG, France*
[c] *Department of Computer Engineering, University of Balamand, Lebanon*
[d] *Faculty of Engineering I, Lebanese University, Lebanon*

Abstract

This letter presents an FPGA implementation of a fault-tolerant Hopfield Neural Network (HNN). The robustness of this circuit against Single Event Upsets (SEUs) and Single Event Transients (SETs) has been evaluated. Results show the fault tolerance of the proposed design, compared to a previous non fault-tolerant implementation and a solution based on triple modular redundancy (TMR) of a standard HNN design.

1. Introduction and Related Work

The architecture of Artificial Neural Networks (ANNs) [1] is considered intrinsically tolerant to faults. However, in hardware implementations, errors can occur due to two types of faults: manufacturing faults such as *stuck-at*'s and non destructive faults (SEUs and SETs) provoked by the impact of energetic particles. These faults, gathered under the name of SEEs (Single Event Effects) must be considered for any circuit or system, even at ground level [2].

Hopfield Neural Networks (HNNs) [3], are a well-known type of ANNs that are able to retrieve an input pattern alrady learnt even if only part of it is available. This powerful concept can be utilized in many applications such as image reconstruction, control, robotics, signal processing, data classification, noise removal, and information retrieval. Field Programmable Gate Arrays

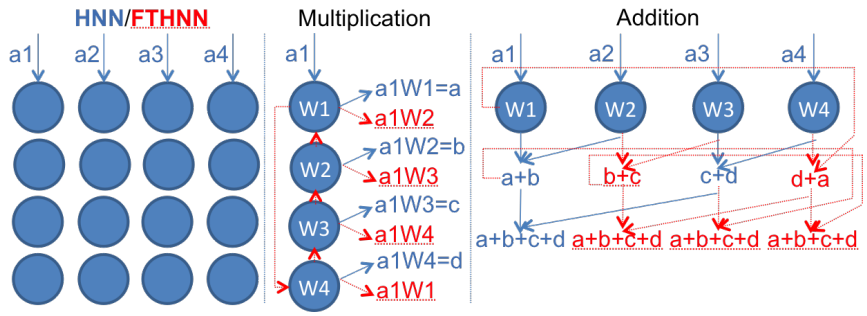

Figure 1: FTHNN architecture. In blue, the elements in the standard HNN and in red/dotted underlined, those of the FT-HNN one. The vector a is the input pattern and w is the matrix of weights of the network

(FPGAs) allow the implementation of very large high-speed ANNs [4, 5]. HNNs have been used for space applications, not only for satellites [6], but also for other hazardous environments [7]. Thus, they can be a target for faults provoked by highly energetic particles. Single Event Upsets (SEUs), and Single Event Transients (SETs) are the most frequent errors provoked by this phenomenon [2]. Hence, their fault tolerance must be analyzed and improved.

This letter presents a hardware implementation on FPGAs of a Fault-Tolerant HNN (FT-HNN) and an experimental study of its robustness against SEUs, SETs and *stuck-at*'s. Faults have been injected with two fault-injection tools: NETFI [8], which emulates SEUs and SETs in the logic of the circuit itself; and NESSY [9], which emulates SEUs into the configuration memory of the FPGA. The robustness of this design is compared with a standard HNN previously developed [5] and a solution based on Triple Modular Redundancy (TMR).

2. Hardware Implementation of the Fault-tolerant HNN

It is not the objective of this letter to describe the architecture of HNNs, since it can be found in [5]. The FT-HNN aims at hardening the adders and multipliers of the HNN. Temporal redundancy is used for multiplications, whereas spatial redundancy (in the order of n) is adopted for additions (Figure 1).

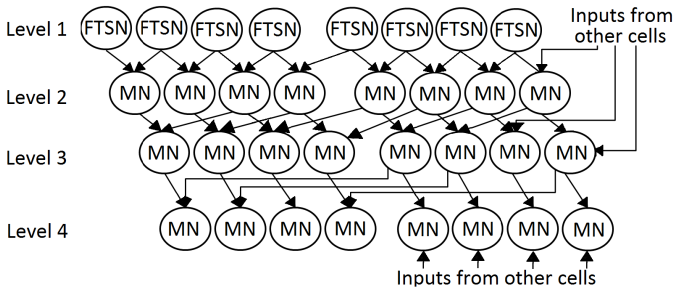

Figure 2: Architecture of a row of 8 nodes in the FT-HNN

Two types of cells exist in the FT-HNN. The first one, named *Fault Tolerant Serial Node* (*FTSN*), multiplies the weights by the input patterns, and adds the result with a serial input coming from other FTSN. FTSNs are used only in the first step of the addition process. The second cell, *Master Node* (*MN*), is used in other steps to add the output of MNs that are at a distance of $2 * n$ cells. Time redundancy is added in the FTSNs with respect to Serial Nodes (*SN*s) depicted in [5]. The latter are not shown in this letter for space reasons.

In our previous work [5] *MN*'s were used to add a pair of nodes. For the FT-HNN (Figure 2), $n * log_2(n)$ master nodes are needed for a row of n nodes. MNs in Level 1 add two consecutive FTSNs, whereas in following levels they add two MNs located $2 * m$ nodes away, m being the level of the MN. A finite state machine (FSM) controls the number of iterations performed.

3. Experimental Results

Figure 3 shows the resource consumption of the three studied versions of the HNN when implemented on two different Xilinx^{TM}FPGAs, which were used used to implement NETFI [8] and NESSY [9], respectively. On average, the resource consumption of the FT-HNN increases by 50.8% the standard one (on average). However, this is very far from the >300% achieved by HNN+TMR. In addition, the maximum operating frequency of the FT-HNN was 398 MHz; whereas for the standard one, it was 439 MHz.

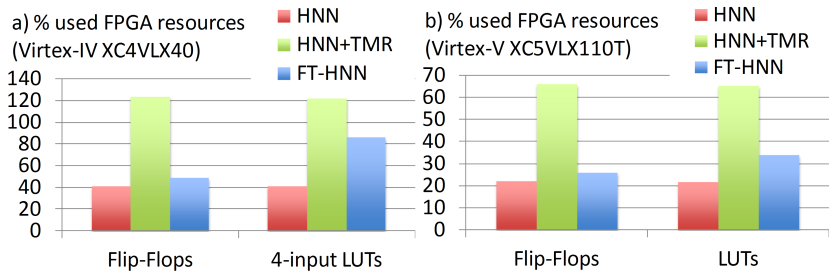

Figure 3: Resource consumption of the three studied HNN versions

Table 1: Results of fault injection using NETFI

	Type of Faults	[#] Faults	Results		
			Errors	Timeouts	Convergences
HNN	SEU	81,831	79 (0.10%)	1,658 (2.26%)	10,865 (13.28%)
	SET	125,676	110 (0.09%)	2,623 (2.09%)	15,195 (12.09%)
	Stuck-at-0	190,509	9,566 (5.02%)	8,417 (4.42%)	19,393 (10.18%)
	Stuck-at-1	98,316	5,444 (5.54%)	3,076 (3.13%)	7,659 (7.79%)
FT-HNN	SEU	137,801	20 (0.01%)	6 (0.004%)	4,789 (3.48%)
	SET	149,198	44 (0.03%)	4 (0.002%)	4,193 (2.81%)
	Stuck-at-0	140,516	3,897 (2.77%)	557 (0.40%)	1,667 (1.19%)
	Stuck-at-1	151,803	4,318 (2.84%)	337 (0.22%)	2,486 (1.63%)

3.1. Study of the SEU and SET Sensitivity at the RTL Level

One fault per execution was injected using NETFI [8]. Consequences are classified as follows: *Silent*, when the fault has no effect on the result; *Error*, where the outputs of the HNN were not the expected ones; *Timeout*, when after a large number of cycles, it does not return any result; and *Convergence*, when it returns correct results, but after the expected execution time.

An extensive fault injection campaign was performed on both versions of the HNN [5]. The obtained SEU and SET error rates drastically decrease for the FT-HNN (see Table 1). Indeed, the faulty results (Errors + Timeouts) decreased from 2.36% to 0.14% in the case of SEUs, from 2.18% to 0.032% for SETs, from 9.44% to 3.07% for *stuck-at-0*'s and from 8.67% to 3.06% for *stuck-at-1*'s. Convergences significantly decrease as well. Thus, the cost in terms of hardware and execution time is justified by a higher robustness, especially for

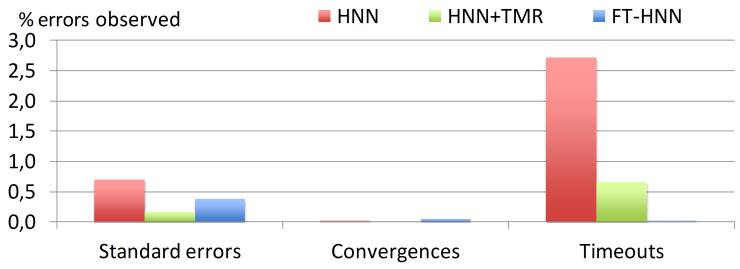

Figure 4: Results of fault injection experiments using NESSY

SEUs. Deeper analyses of the results show that the global FSM is the most sensitive part of the design.

3.2. SEU Sensitivity of the FPGA Configuration Memory

By using NESSY [9], SEUs were also injected in all the configuration bits used for implementation of the HNNs (Figure 4). As expected, the SEU sensitivity of the HNN+TMR significantly decreases with respect to the standard one (-76% on average). The FT-HNN also improves the standard one. In comparison with the HNN+TMR, the sensitivity of the FT-HNN increases to 0.386% in case of errors, but it almost reduces the number of timeouts to zero (0.027%). Convergences slightly increase as well, from 0.024% to 0.053%. However, in general terms (errors + timeouts + convergences), the FT-HNN hardens the HNN by a factor of 7.41 (0.16% vs. 1.15%) and HNN+TMR, by a factor of 2.47 (0.16% vs. 0.28%). Thus, the most interesting conclusion that can be drawn is that the FT-HNN introduces an affordable resources overhead (+50.8%), but in exchange, it is 7.41 times more robust to SEUs. This is especially interesting in comparison with the HNN+TMR (which featured +>300% hardware resources overhead, and in exchange, only a factor of SEU sensitivity reduction of 2.47).

Acknowledgements

This work has been supported by the Spanish Ministry of Education, Culture and Sports under grant TIN2013-40968-P and by the mobility grant for professors and researchers "José Castillejo".

References

[1] D. E. Rumelhart, J. L. McClelland, C. PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, MIT Press, Cambridge, MA, USA, 1986.

[2] E. Normand, Single Event Effects in Avionics, IEEE Transactions on Nuclear Science 43 (2) (1996) 461 – 474.

[3] J. J. Hopfield, Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, USA, 1988, Ch. Neurons with Graded Response Have Collective Computational Properties Like Those of Two-state Neurons, pp. 577 – 583.

[4] M. Stepanova, F. Lin, V. C.-L. Lin, A Hopfield Neural Classifier and its FPGA Implementation for Identification of Symmetrically Structured DNA Motifs, Journal on VLSI Signal Processing Systems 48 (3) (2007) 239 – 254.

[5] W. Mansour, R. Ayoubi, H. Ziade, R. Velazco, W. EL Falou, An Optimal Implementation on FPGA of a Hopfield Neural Network, Advances in Artificial Neural Systems 2011 (2011) 7:1 – 7:9.

[6] Q. Wang, W. Shi, P. Atkinson, Z. Li, Land cover change detection at subpixel resolution with a hopfield neural network, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 8 (3) (2015) 1339 – 1352.

[7] A. Kzar, M. MatJafri, H. Lim, K. Mutter, S. Syahreza, Modified Hopfield Neural Network Algorithm (MHNNA) for TSS mapping in Penang strait, Malaysia, in: IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 2013, pp. 187 – 192.

[8] W. Mansour, R. Velazco, An Automated SEU Fault-Injection Method and Tool for HDL-Based Designs, IEEE Transactions on Nuclear Science 60 (4) (2013) 2728 – 2733.

[9] V. Alaminos, F. Serrano, J. A. Clemente, H. Mecha, NESSY: An Implementation of a Low-Cost Fault-Injection Platform on a Virtex-5 FPGA, in:

the Conference on Radiation and its Effects on Components and Systems (RADECS), 2012.