Protection sélective d'un réseau de neurones implémenté sur puce FPGA soumis à un environnement radiatif

Wilfread GUILLEMÉ[1] Youri HELEN[2] Rémy PRIEM[2] Angeliki KRITIKAKOU[1] Cédric KILLIAN[1] Daniel CHILLET[1]

[1]Université de Rennes, INRIA, IRISA, 263 Av. Général Leclerc, 35000 Rennes, France

[2]DGA MI, 136 La Roche Marguerite, 35170 Bruz, France

Résumé – Les applications avioniques sont confrontées à de nombreux défis pour garantir un fonctionnement sûr et fiable. Elles doivent être capables de fonctionner sous environnements contraints dans lesquels les circuits sont soumis à l'effet de particules ionisantes ou de rayons cosmiques. En outre, ces systèmes requièrent des traitements de plus en plus complexes pouvant, dans certains cas, être avantageusement remplacés par des réseaux de neurones dont l'implémentation sur circuits FPGA peut être très efficace. Afin de répondre à la problématique d'exécution de ces algorithmes au regard des fautes d'origines radiatives, une protection sélective des éléments de stockage du calcul des neurones est proposée. Cette méthode repose sur la triplication des bascules détectées comme sensibles à l'issue d'une simulation. Les résultats de l'étude indiquent que la phase d'apprentissage impacte la sensibilité des différentes couches du réseau de neurones. Les bits de signe et de poids fort qui stockent le résultat des calculs des neurones sont particulièrement vulnérables et doivent être protégés. La stratégie de protection sélective présentée dans cet article permet d'augmenter la tolérance aux fautes de ces systèmes en offrant un compromis entre la consommation de ressources matérielles et la qualité du résultat de prédiction du réseau de neurones en présence de fautes.

Abstract – Avionics applications face many challenges to ensure safe and reliable operation. They must be able to operate under constrained environments in which circuits are subject to the effect of ionizing particles or cosmic rays. In addition, these systems require increasingly complex processing, which has led to the exploration of solutions such as the use of embedded neural networks on FPGA. To address the problem of executing these algorithms with respect to faults of radiative origin, a protection method is proposed. It is based on the triplication of flip-flops detected as sensitive after a simulation. The results of the study indicate that the learning phase impacts the sensitivity of the different layers of a multilayer perceptron. The sign and most significant bits that store the result of the artificial neuron calculations are particularly vulnerable and must be protected. The selective protection strategy presented in this article increases the fault tolerance of these systems by offering a trade-off between hardware resource consumption and neural network prediction quality in the presence of faults.

1 Introduction

Les systèmes embarqués dans les applications critiques doivent faire face à des contraintes pour garantir un fonctionnement sûr et fiable. Ils doivent être en mesure de supporter des environnements difficiles. Les effets des radiations ionisantes sont susceptibles de causer des erreurs dans les circuits électroniques, compromettant ainsi le fonctionnement du système. Un exemple est le Single Event Upset (SEU) [1] qui se caractérise par l'inversion d'un bit contenu dans une bascule. Les architectures électroniques modernes y sont de plus en plus sensibles avec la diminution de la taille des transistors [2]. Cette problématique s'intensifie avec l'augmentation des particules en altitude [3]. Dans ce contexte, la protection des applications embarquées dans un système avionique devient alors un enjeu crucial pour assurer la bonne conduite des missions qui reposent en partie sur des algorithmes de reconnaissance d'objets ou de contrôle de vol. L'évolution de la complexité de ces algorithmes a progressivement conduit à se tourner vers l'implémentation de réseaux de neurones artificiels dont il est reconnu qu'ils sont par nature résilients aux fautes et aux approximations de calcul [4]. Toutefois cette résilience n'est pas totale et un SEU peut entraîner une mauvaise décision du système en raison d'une erreur de prédiction du modèle.

L'implémentation d'un réseau de neurones sur FPGA présente de nombreux avantages, notamment grâce à la possibilité de réaliser des calculs en parallèle, accélérant ainsi le temps de calcul. De plus, la conception sur mesure de l'architecture embarquée pour une application spécifique permet de réaliser des calculs complexes avec une efficacité énergétique satisfaisante. La technologie Flash est généralement préférable à la technologie SRAM car la configuration est stockée dans des mémoires non volatiles, qui sont moins sensibles aux effets des radiations. Néanmoins, afin d'améliorer le niveau de durcissement du système électronique, il est possible de le protéger des fautes grâce à différentes techniques telles que la triplication (TMR) [5] ou la redondance d'informations par les codes correcteurs d'erreurs Hamming [6] et BCH [7]. Ces techniques peuvent être coûteuse alors que les propriétés de résilience des réseaux de neurones permettent d'envisager uniquement la protection des éléments les plus sensibles du système et ainsi d'économiser des ressources matérielles.

Cet article propose d'adresser cette problématique, nous présentons une méthode de protection sélective d'un perceptron multicouche [8] permettant la reconnaissance de chiffres. Ce réseau de neurones a été développé en Python puis implémenté en VHDL pour ensuite être instancié sur une cible FPGA. Dans un premier temps, l'architecture du réseau de neurones est décrite et le scénario de test permettant de détecter les bascules sensibles est introduit. Les résultats obtenus sont justifiés par

des études portant sur la résilience de l'algorithme en fonction du nombre d'itérations de la phase d'apprentissage, la vulnérabilité des bascules des neurones et la sensibilité des différentes couches. Enfin, une estimation des ressources requises pour protéger l'architecture est proposée.

2 Architecture du réseau de neurones

Cette section présente l'architecture du perceptron multicouche étudié. Elle expose le contexte de la phase d'apprentissage ainsi que les différences qui existent entre la conception du modèle et son implémentation matérielle.

2.1 Phase d'apprentissage

Le réseau de neurones étudié est un perceptron multicouche dont l'objectif est de reconnaître des chiffres. Il a été décrit à l'aide de Keras/TensorFlow [9] et entraîné à partir de la base de données MNIST [10]. Ce *dataset* est constitué de 70 000 images de chiffres manuscrits, dont 60 000 sont destinées à l'entraînement du modèle et 10 000 sont utilisées pour l'inférence et la validation. Ces images, à l'origine de taille 28x28 en nuances de gris ont été modifiées en images 8x8 en noir et blanc, elles constituent les 64 entrées du réseau de neurones.

2.2 Architecture du modèle

La figure 1 présente l'architecture du perceptron multicouche étudié, elle montre que la couche d'entrée du modèle est de taille 64, qu'il est constitué d'une couche intermédiaire avec 64 neurones ainsi qu'une couche de sortie avec 10 neurones donnant la probabilité d'appartenance aux 10 classes possibles.

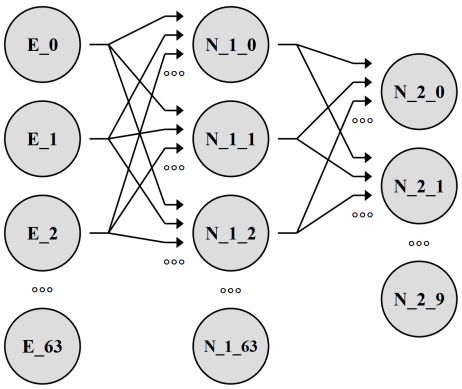

FIGURE 1 : Architecture du perceptron multicouche.

La fonctionnalité d'un neurone artificiel, comme le montre la figure 2 repose sur un bloc de logique combinatoire qui effectue l'opération multiplication accumulation entre les entrées X_i et les poids W_i, un biais B est ajouté. Le résultat est ensuite stocké dans un registre (composé de bascules) avant d'être envoyé vers la mémoire qui modélise la fonction d'activation sigmoïde.

2.3 Représentation des données sur FPGA

Les données et les paramètres d'un réseau de neurones sous l'environnement Python sont des nombres réels et signés. Ces valeurs sont instanciées dans le format à virgule flottante

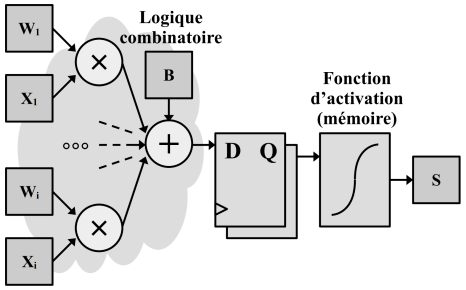

FIGURE 2 : Schéma fonctionnel d'un neurone artificiel.

simple ou double précision, codées respectivement sur 32 bits et 64 bits. Cet encodage n'est pas optimal pour les FPGA. En effet, la ressource matérielle y est beaucoup plus limitée. Dans l'environnement FPGA, les données sont représentées par le type SFIXED qui permet de représenter un nombre réel signé et d'obtenir un format de la donnée sur-mesure. Le bit de poids fort correspond au signe, les bits suivants correspondent à la partie entière puis à la partie décimale avec une taille indiquée. L'encodage SFIXED retenu dans notre cas, illustré à la figure 3, est constitué de 10 bits dont 1 pour le signe, 4 pour la partie entière et 5 pour la partie décimale.

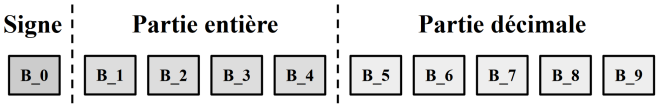

FIGURE 3 : Format de données SFIXED retenu pour notre application.

3 Détection des bascules sensibles

Cette section présente la solution matérielle permettant l'injection d'une faute au sein d'un neurone, l'architecture du système permettant la détection des bascules sensibles d'un perceptron multicouche ainsi que les données d'entrée utilisées.

3.1 Injection d'une faute au sein d'un neurone

Afin de détecter les bascules sensibles de l'architecture, une solution permettant d'émuler un SEU au niveau des neurones doit être développée comme indiqué dans la figure 4. Cette solution repose sur l'ajout de signaux supplémentaires permettant l'inversion du contenu d'une bascule. Un multiplexeur commandé par le signal "Injection_faute" est utilisé pour sélectionner la bascule à tester, tandis que des portes inverseuses permettent de changer le contenu de cette dernière. Grâce à cette méthode, la sensibilité aux fautes de toutes les bascules du réseau de neurones peut être évaluée.

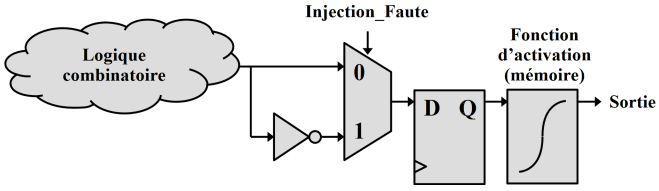

FIGURE 4 : Injection d'une faute dans un neurone artificiel.

3.2 Architecture matérielle de la détection

L'implémentation matérielle de l'architecture de détection de faute est présentée en figure 5. Elle est constituée de deux réseaux de neurones identiques exécutés en parallèle, ils sont notés ANN1 et ANN2. Le premier modèle fonctionne normalement tandis que le second modèle est soumis au dispositif d'injection de fautes présenté dans la section 3.1 capable de simuler un SEU sur toutes les bascules des neurones.

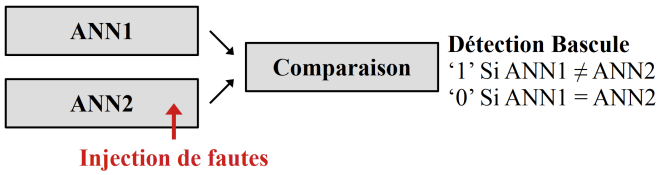

FIGURE 5 : Implémentation matérielle de la détection des bascules sensibles.

Les résultats de prédiction des deux perceptrons multicouches sont ensuite comparés. Si les prédictions fournies par les modèles pour toutes les combinaisons d'entrées possibles sont identiques, cela signifie que la bascule sélectionnée n'a pas provoqué d'erreur de prédiction du modèle. Elle n'est donc pas considérée comme sensible. À l'inverse, si les sorties des deux réseaux de neurones sont différentes pour des données d'entrée identiques, cela signifie que la bascule désignée a provoqué une erreur de prédiction du modèle, elle est donc considérée comme sensible.

La procédure de test traduite par l'algorithme 1 permet d'inverser successivement le contenu des 10 bascules de chacun des 74 neurones pour différentes images d'entrées. L'exécution de cet algorithme permet de dresser une table de sensibilité des bascules. Cette méthode offre la possibilité de définir un niveau de sensibilité pour chacun des éléments séquentiels. Par exemple, une bascule détectée comme sensible sur toutes les images du scénario de test obtiendra un niveau de sensibilité de 1. Une bascule détectée sensible sur la moitié des images aura quant à elle un niveau de sensibilité de 0,5. Ces différentes valeurs aideront à choisir les bascules prioritairement à protéger en privilégiant celles ayant le plus grand niveau de sensibilité.

Algorithme 1 : Scénario de test de détection des bascules sensibles.

```
1  i = 0 (Selection de l'image)
2  n = 0 (Selection du neurone)
3  b = 0 (Selection de la bascule)
4  pour i = 0...NB_Images faire
5      pour n = 0...NB_Neurones faire
6          pour b = 0...NB_Bascules faire
7              ANN1 = CalculANN(i);
8              InversionBascule(n, b);
9              ANN2 = CalculANN(i);
10             InversionBascule(n, b);
11             si ANN1 ≠ ANN2 alors
12                 BasculeSensible(n, b) + +
13             fin
14         fin
15     fin
16 fin
```

4 Résultats obtenus

Cette section présente les résultats obtenus au terme du scénario de détection des bascules sensibles. Les études portent sur la vulnérabilité des bascules des neurones et la sensibilité des différentes couches. Les variations de la résilience de l'algorithme selon le nombre d'itérations de la phase d'apprentissage sont analysées. Enfin, la consommation de ressources en bascules de la méthode de protection proposée est évaluée.

4.1 Sensibilité des couches et de la donnée

Le tableau 1 est obtenu à l'issue du scénario de test de détection des bascules sensibles présenté à la section précédente. Il s'agit d'un modèle ayant effectué 1000 itérations d'apprentissage avec un *batch_size* de 32.

	Signe	Partie entière				Partie décimale				
Neurone	B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
N_1_0	0.1	0.05	0.1	0	0.05	0.05	0	0	0	0
N_1_1	0.2	0	0.05	0	0	0	0	0	0	0
N_1_2	0.25	0.05	0	0	0	0	0	0	0	0
N_1_3	0.1	0.05	0.05	0.05	0	0	0	0	0	0
...										
N_1_63	0.1	0.1	0.05	0	0	0	0	0	0	0
N_2_0	0.55	0	0.05	0	0	0	0	0	0	0
N_2_1	0.4	0.1	0.1	0	0.1	0	0	0	0	0
N_2_2	1	0.05	0.1	0	0	0	0	0	0	0
N_2_3	0.9	0.1	0.15	0.05	0.05	0	0	0	0	0
N_2_4	0.6	0.1	0.05	0	0	0	0	0	0	0
N_2_5	0.95	0.1	0.05	0.05	0.05	0.05	0	0	0	0
N_2_6	0.5	0.05	0.05	0.05	0	0	0	0	0	0
N_2_7	0.8	0.05	0	0.1	0.1	0	0	0	0	0
N_2_8	0.85	0.15	0.1	0.1	0.1	0.05	0	0	0	0
N_2_9	1	0	0.1	0	0	0	0	0	0	0

TABLEAU 1 : Niveau de sensibilité des différentes bascules.

Le tableau 1 montre que les bascules contenant les bits de signe sont les plus sensibles ce qui est sans surprise puisque l'inversion de ce bit modifie fortement la valeur codée. Dans une moindre mesure, on remarque que les bits de poids forts peuvent également avoir un impact sur le résultat puisqu'ils peuvent changer fortement la donnée d'entrée de la fonction d'activation. Cette étude montre également que la couche de sortie est beaucoup plus sensible que la couche intermédiaire. Ceci peut s'expliquer par un plus faible nombre de neurones constituant cette couche définissant le résultat de prédiction du modèle.

4.2 Résilience en fonction de l'apprentissage

Au cours de ce travail, plusieurs réseaux de neurones ont été générés. Leur tolérance aux fautes varie à cause de la nature aléatoire de la phase d'apprentissage. Pour cette raison, une étude de la tolérance aux fautes en fonction du nombre d'itérations d'apprentissage a été menée. Les résultats de cette étude, présentés dans la figure 6, mettent en évidence deux évolutions distinctes du nombre de fautes pouvant provoquer une prédiction incorrecte du réseau de neurones. La courbe bleue trace les résultats de la détection de fautes lors de l'utilisation des images d'entraînement, tandis que la courbe rouge représente les résultats lorsque les images de tests sont appliquées au réseau. Les paramètres des modèles ont été extraits à différents moments de l'apprentissage, qui sont représentés sur l'axe des abscisses. Les résultats indiquent que la tolérance aux fautes du réseau de neurones augmente avec le nombre d'itérations d'apprentissage. En outre, le modèle est plus résilient lorsqu'il est exposé à un environnement pour lequel il a été conçu. Le réseaux de neurones est plus tolerant aux

fautes sur la base d'images d'entraînement que sur la base de test/validation. Pour ce résultat, le modèle de réseau de neurones le plus résilient est celui ayant effectué 1000 itérations d'apprentissage.

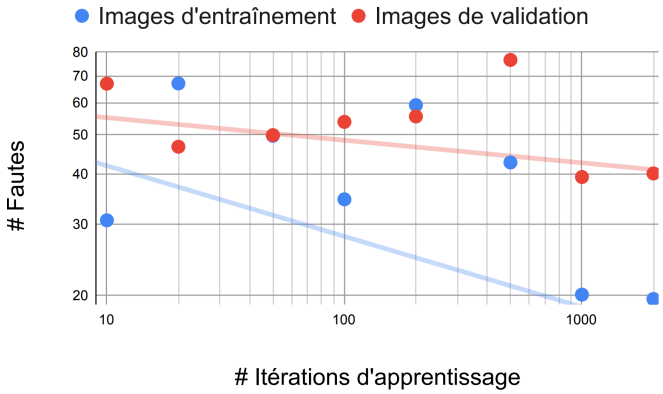

FIGURE 6 : Nombre de fautes détectées par le scénario de test.

4.3 Consommation de ressources matérielles

La figure 7 illustre le nombre de bascules requis pour implémenter le réseau de neurones étudié, selon les stratégies de protection et les itérations d'apprentissage. Avec 74 neurones artificiels composés chacun de 10 bascules, le système non protégé nécessite 740 bascules. Alors qu'une protection totale de toutes les bascules par triplication classique nécessiterait 2220 bascules, la méthode proposée réduit le nombre de bascules nécessaires à 1200 pour le modèle ayant subi 1000 itérations d'apprentissage, soit une diminution de 45% par rapport à la méthode de triplication classique, tout en offrant une protection similaire.

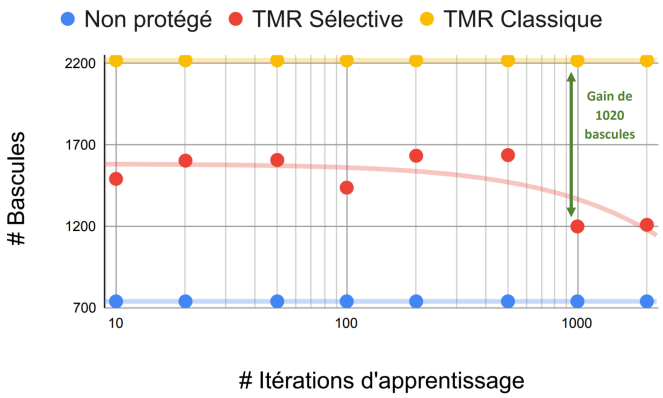

FIGURE 7 : Coût en bascules des méthodes de durcissement.

5 Conclusion

Cet article présente une méthode de protection sélective d'un réseau de neurones implémenté sur FPGA. Une méthodologie de test a été développée pour détecter les bascules sensibles et une solution matérielle a été proposée pour la mise en œuvre de ce scénario. Les résultats obtenus montrent que les vulnérabilités aux fautes des couches d'un réseau de neurones ne sont pas identiques, la dernière couche est la plus sensible. De plus, l'apprentissage joue un rôle important dans l'amélioration de la tolérance aux fautes intrinsèques du réseau de neurones. Augmenter l'apprentissage permet d'améliorer la tolérance aux fautes de l'architecture. Enfin, la méthode proposée dans cet article permet de réduire considérablement le coût matériel de la protection de l'implémentation du réseau de neurones tout en offrant une protection se rapprochant d'une triplication classique.

6 Remerciements

Ce travail est soutenu par la Direction Générale de l'Armement (DGA), financé par l'Agence de l'Innovation de Défenses (AID) et l'institut national de recherche en sciences et technologies du numérique (INRIA).

Références

[1] Paul E Dodd and Lloyd W Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *IEEE Transactions on nuclear Science*, 50(3) :583–602, 2003.

[2] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. The impact of technology scaling on lifetime reliability. In *International Conference on Dependable Systems and Networks, 2004*, pages 177–186. IEEE, 2004.

[3] Space Product Assurance. Techniques for radiation effects mitigation in aasic and fpgas handbook. Technical report, Technical Report. ESA Requirements and Standards Division, 2016.

[4] Alberto Bosio, Paolo Bernardi, Annachiara Ruospo, and Ernesto Sanchez. A reliability analysis of a deep neural network. In *2019 IEEE Latin American Test Symposium (LATS)*, pages 1–6, 2019.

[5] Melanie Berg and Kenneth A LaBel. Verification of triple modular redundancy (tmr) insertion for reliable and trusted systems. In *2016 MRQW Microelectronics Reliability and Qualification Working Meeting*, number GSFC-E-DAA-TN29375, 2016.

[6] K Furutani, K Arimoto, H Miyamoto, T Kobayashi, K Yasuda, and K Mashiko. A built-in hamming code ecc circuit for drams. *IEEE Journal of Solid-State Circuits*, 24(1) :50–56, 1989.

[7] Robert Chien. Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes. *IEEE Journal on information theory*, 10(4) :357–363, 1964.

[8] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15) :2627–2636, 1998.

[9] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[10] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6) :141–142, 2012.