Reliable and Efficient hardware for Trustworthy Deep Neural Networks

# Reliable and Efficient hardware for Trustworthy Deep Neural Networks

Salvatore Pappalardo[1], Ali Piri[1], Annachiara Ruospo[2],
Bastien Deveautour[3], Ernesto Sanchez[2], Alberto Bosio[1]

[1]*Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, 69130 Ecully, France*
[2]*Politecnico di Torino, Dip. di Automatica e Informatica, Torino, Italy*
[3]*Nantes Université, CNRS, IETR UMR 6164, Nantes, France*
Email: [1]{name.surname}@ec-lyon.fr, [2]{name.surname}@polito.it, [3]{name.surname}@univ-nantes.fr

***Abstract*—The remarkable capacities of modern DNNs facilitate their use in real-time, edge and mobile applications, as well as in safety-critical systems (e.g., autonomous cars). In these contexts, very opposite goals need to be achieved simultaneously: performance, energy consumption and reliability. Hardware accelerators can be an alternative when targeting the first two goals, however, they usually require redundancy techniques to reach the expected DNN resilience level, negatively impacting development costs in terms of area, power and latency. This work explores the use of approximate computing to increase DNN resilience without resorting to redundancy techniques, thereby avoiding additional costs.**

***Index Terms*—approximate computing, deep neural networks, systolic array, reliability.**

## I. Introduction

Deep Neural Networks (DNNs) have revolutionized the field of artificial intelligence and machine learning, due to their capacity to learn from vast amounts of data, making very complex predictions. Nowadays, the industry is employing them in many different fields including safety-critical ones, such as autonomous driving systems [1].

Some of the main challenges related to DNNs utilization, regard their high energy consumption, substantial computational power requirements, and long inference times to complete a single task.

To tackle these issues, new types of hardware accelerators have been developed. Systolic arrays are among the top choices, which have gained considerable popularity in the last years. They are usually composed by a grid of Processing Elements (PEs), where each PE performs a Multiply and Accumulate (MAC) operation, meanwhile forwarding the data to the neighboring PEs. The PE grids are able to process a convolution with $O(n \times m)$ time complexity (supposing a convolution between two matrices with sizes $n \times n$ and $m \times m$). Due to their low energy consumption requirements, they can be used in embedded systems [1].

On the other hand, it has been proven that hardware faults affecting hardware accelerators can significantly affect the inference, leading to DNN prediction failures that are likely to lead to a detrimental effect on the application [2]–[4].

979-8-3503-6312-8/24/$31.00 ©2024 IEEE

Therefore, ensuring the resilience of hardware accelerators is crucial, particularly when they are deployed on real-time safety-critical systems. These systems carry a high level of risk, as any failure may have life-threatening consequences [5].

In order to improve the system resilience, many fault tolerance mechanisms can be adopted. A typical solution consists on adding redundancy to the original design [5]. However, the problem of using redundancy, even in the case of selective fault tolerance techniques, is the increasing costs in terms of area, power and latency that can be prohibitive, especially for edge applications. Therefore, there is a need to **further reduce the fault tolerance costs** for DNN hardware accelerators.

This research work proposes a novel approach entirely based on the use of Approximate Computing (AxC) to increase the resilience of DNNs, eliminating the necessity to resort to redundancy techniques and thus avoiding extra costs. In particular, an analysis to understand the impact in the resilience of the DNN is made resorting to AxC techniques based on bit-width reduction and functional approximation of the multipliers included in a systolic array architecture used to accelerate the DNN execution. In our case study, a fault injection (FI) campaign is conducted by injecting single bit-flips in the PEs of the systolic array, gathering information to determine the resilience of the network. Results have shown that by including AxC modules in the DNN, it is possible to **reduce 8 times** the percentage of critical faults with respect to the accurate implementation, increasing the network resiliency while guaranteeing better efficiency in the final design.

The rest of this paper is structured as follows: in section II a brief overview of existing works is given. Section III presents the systolic array architectures. Section V shows the gathered results; section VI concludes the article, highlighting future directions.

## II. Related works

Several works have been published so far concerning the use of AxC in the context of fault tolerance. This research area is generally known as Approximate Triple Modular Redundancy , applied at the circuit level. This approach employs three Approximate Integrated Circuits instead of three fully-precise replicas. For a given input, only one circuit can provide an incorrect answer. However, the fault tolerance capability mainly depends on the voter that has to be modified [5].

A different approach referred to as Quadruple Approximate Modular Redundancy was presented [6]. It is based on the idea of selectively approximating a subset of the circuit. The final goal is to have four approximate replicas in a manner that ensures that, out of the four, at least three will provide precise results for a given output. The benefit is that the voter does not have to be modified. However, the design cost is higher since the approximation must be carefully identified.

The described works target fault tolerance through masking. Concerning fault detection only, in [7], the authors exploit approximate computing for image processing applications through duplication with comparison .

The above techniques summarize the use of AxC to reduce the cost of testing and fault detection/tolerance. Another interesting branch of research is investigating how AxC impacts a system's intrinsic reliability, and whether an approximate application is more or less resilient to hardware faults than the precise application. Indeed, AxC exploits the inherent resilience of an application to noise and computing errors. Therefore, an application executed on approximated hardware is less resilient to hardware faults.

In [8], the authors studied the impact of AxC on the reliability of DNN application. As AxC technique, they used the data type and bit-width reduction. In particular, they compared 32-bit floating-point versus 16-bit and 8-bit integer data for storing synaptic weights. The DNN was an image classifier based on LeNet-5 topology. The reliability assessment has been done through radiation experiments: the system running the DNN was exposed to neutron beam.

In [2], [3], the authors proposed a specific fault-tolerant technique for systolic arrays used to deploy DNNs. In [2], permanent faults are targeted, and the idea consists in skipping the computation of a faulty PE. Contrarily, [3] targets transient faults and the approach consists in computing the same operation in two PEs and comparing the results. If they differ, the result is not fed to the next PEs. Both techniques correspond to prune the DNN.

This work exploits two different AxC techniques in systolic arrays: bit-width reduction (concerning weights) and approximate multipliers. It will be shown that it is possible to increase resilience of DNNs **without leveraging redundancy**.

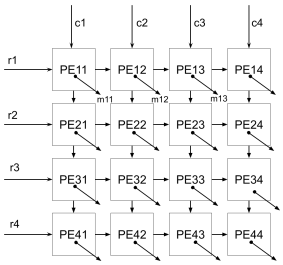
## III. Systolic Array

As [9] points out, there are three main types of systolic arrays: **(i) Weight stationary**, in which the weights are "fixed" at each PE and the accumulation operation is performed through the columns; **(ii) Output stationary**, in which both weights and activations flow in the array in perpendicular directions, while the Processing Elements (PEs) perform the MAC operation using an accumulation register; **(iii) No local reuse**, similarly to the first type, the partial sum flows through the columns, but in this case the weights have dedicated forwarding lines.

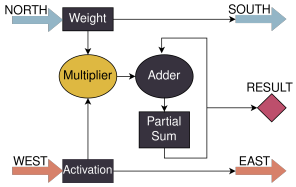
We developed a custom systolic array of the output stationary type. Figure 1a shows the general architecture. The weights flow "vertically" (i.e., from the NORTH to the SOUTH of each PE) while the activations flow "horizontally". Figure 1b shows a functional diagram of a PE. As above-mentioned, a PE performs a MAC operation; this means that for each clock cycle:

- `north` and `west` inputs are multiplied together,
- the result is added to the partial sum register,
- the value of `north` is put on the `south` output,
- the value of `west` is put in the `east` output.

These operations accumulate a series of multiplications and forward data to the neighbors. In this implementation, weights are flowing from north to south and the activations from west to east and are accumulated in the PE itself.




(a) Custom systolic array




(b) Single PE internals

Fig. 1: Systolic array architecture

In the upcoming subsections, we will elaborate on the specific approximation techniques we employed and their application to the systolic array.

### A. Bit-width reduction

The bit-width reduction is a classical approximation technique [5] for which the size of the data manipulated by the application is reduced w.r.t. the precise implementation. In the context of DNNs, the bit-width reduction is better known as quantization since the DNN parameters (i.e., weights, bias, ...) are "converted" from the 32-bit floating-point representation to $n-bit$ integer representations. From the energy point of view, the bit-width reduction allows reducing memory footprint and data movement. The systolic array has been implemented with two data representations: the INT 16-bit as the reference (i.e., precise) and the INT 8-bit as the approximated precision.

### B. Functional approximation

Functional approximation [10] aims at modifying the circuit structure in order to obtain a simpler implementation (i.e., less costly) with a reduced precision compared to the precise implementation. This technique has been widely adopted with

arithmetic circuits (i.e., adders and multipliers), obtaining benefits in terms of area, power, and latency reduction. In this work, experimental analyses have been conducted with different public available approximate multipliers from the EvoApproxLiteLITE library[1]

During DNN inference, multiplications involve as operands signed weights and unsigned activations, necessitating the use of a mixed-sign multiplier, which is not available in the aforementioned library. Therefore, we configured experiments using a 12-bit signed multiplier and simply extending the sign of the operands. Note that this method worsens the approximation: since the multipliers are designed to operate on 12-bits values, using only the least significant 8 bits means reducing their performance. To quantify this effect, we computed the Mean Absolute Error (MAE) and the Worst Case Error (WCE) with the mixed-sign range the networks actually use. Table I shows the *original* metrics of the multiplier (columns MAE, WCE) and the newly computed metrics that only use our 8-bit values ranges (columns MAE-8, WCE-8).

TABLE I: Characterization of the approximate multipliers.

| Name | MAE % | WCE % | MAE-8 % | WCE-8 % |
|---|---|---|---|---|
| mul12s_2PT | 0.000073 | 0.00029 | 0.019 | 0.0748 |
| mul12s_2QH | 0.0031 | 0.013 | 0.134 | 0.514 |
| mul12s_2R5 | 0.0092 | 0.037 | 0.315 | 1.234 |
| mul12s_34P | 0.032 | 0.17 | 0.785 | 3.920 |
| mul12s_2TE | 0.19 | 0.77 | 6.080 | 24.417 |

## IV. Resilience Analysis

The resilience analysis is based on the framework published in [11]. We injected transient faults (i.e., modeled as single bit-flip) affecting the inputs of PEs in the systolic array. The faults' universe is then 3-dimensional, and its size depends on both the array size and the bit-width:

$$\text{FaultSpace} = 2 \times K \times M \tag{1}$$

where $M$ is the number of systolic array's PEs, and $K$ is the bit-width of the PE (i.e., 8- or 16-bit). Please note that we injected faults in the PE input corresponding to the weights of the DNN. Each fault corresponds to an alteration of one bit, forcing its value to either '1' or '0'.

A statistical FI was configured by assuming a 1% margin of error, 50% probability of a fault resulting in a failure, a cut-off point of 2.58 which corresponds to 99% confidence level [12].

For each input, two outputs are collected: the fault-free $\hat{Y}$ and the faulty output $\tilde{Y}$. They are vectors of 10 components, each corresponding to the probability of a class. The classification labels $\hat{y}$ and $\tilde{y}$ are obtained as $\hat{y} = argmax(\hat{Y})$ and $\tilde{y} = argmax(\tilde{Y})$. For the purpose of the resilience classification, $\hat{Y}$ and $\tilde{Y}$ are compared for each pair input-fault as follows:

- we call a fault **Masked** when $\hat{Y} = \tilde{Y}$,
- **Tolerable** when
  - $\hat{y} = \tilde{y}$ and $\frac{max(\tilde{Y})}{max(\hat{Y})} > 1$,
  - $\hat{y} = \tilde{y}$ and $0.95 < \frac{max(\tilde{Y})}{max(\hat{Y})} < 1$,
- **Critical** when
  - $\hat{y} = \tilde{y}$ and $\frac{max(\tilde{Y})}{max(\hat{Y})} < 0.95$,
  - $\hat{y} \neq \tilde{y}$.

Basically, we check whether the top-1 probability of the injected network $max(\tilde{Y})$ is greater or smaller than the golden top-1 probability $max(\hat{Y})$, and classify the fault accordingly. Note that **masked** faults do not produce any difference in the output, while only **critical** faults involve misclassification.

## V. Experiments

To validate the effectiveness of the technique, the LeNet-5 CNN architecture, trained and tested on MNIST dataset, has been exploited. It is composed of three convolutional layers and four fully connected. The training process was performed by using the accurate version of the network without any approximation; the training parameters are: learning rate started at 0.05, with the decay of $5 \times 10^{-4}$ every 375(*128) iterations, the momentum was set to 0.9. The final accuracy of the model was equal to 99.05%. We then performed 16- and 8-bit quantization through the following steps.

1) all weights are rescaled in the range $[-1.0, 1.0]$ and activations at each layer are rescaled in the range $[-1.0, 1.0]$ for signed outputs and $[0.0, 1.0]$ for unsigned outputs;
2) inputs, weights, biases and activations are quantized to the desired $n_{bits}$ by converting $[-1.0, 1.0]$ and $[0.0, 1.0]$ to $[-2^{n_{bits}-1} - 1, 2^{n_{bits}-1} - 1]$ and $[0, 2^{n_{bits}-1} - 1]$

Table II reports the accuracy of the quantized networks.

The target CNN is then deployed in our systolic array, sized as $28 \times 28$ in the configurations depicted in Table II. The Table reports the different systolic array configurations (numbered from 1 to 6 in the first column). Each configuration is characterized by its bit-width precision (16/8 bits) and the type of approximate multiplier (second and third columns). Columns four and five report the performance in terms of energy reduction w.r.t the baseline, and the classification accuracy. The last columns show the results of the FI campaign expressed in terms of fault percentage of injected faults w.r.t. the total one [12], and the FI outcomes (i.e., percentage of Masked, Tolerable and Critical faults) found in every fault injection campaign.

As expected, the approximation has a positive impact on the energy reduction at a very limited accuracy degradation except for the configuration #6, for which the multiplier has a very low precision. The interesting observation is the impact on the resilience, here represented as injection outcomes, of the approximation. Firstly, the simple use of bit-width reduction reduces by a factor of 2 the amount of critical faults. Second, the approximate multiplier significantly impacts the resilience and the impact depends on the adopted multiplier. We can cite the case #5 for which we reduced by **8x** the amount of critical faults **without adding any redundancy mechanisms**.

We further investigated why we obtained such improvement for the case #5. The intuition behind such behavior leverages

[1]https://ehw.fit.vutbr.cz/evoapproxlib/?folder=multipliers/12x12_signed

TABLE II: Summary of Experimental Results.

| # | Bit-width | Multiplier | Energy Reduction [%] | Accuracy [%] | Injected Faults [%] | Masked [%] | Tolerable [%] | Critical [%] |
|---|---|---|---|---|---|---|---|---|
| Baseline | 16 | Precise | – | 99.07% | 10% | 47.58% | 29,18% | 23.24% |
| 1 | 8 | Precise | 50% | 99.05% | 19% | 64.65% | 23.01% | 12.34% |
| 2 | 8 | mul12s_2PT | 50.3% | 99.08% | 19% | 63.9% | 23.79% | 12.3% |
| 3 | 8 | mul12s_2QH | 51.21% | 99.1% | 19% | 38.92% | 44.85% | 16.23% |
| 4 | 8 | mul12s_2R5 | 52% | 99.06% | 19% | 26.96% | 55.69% | 17.34% |
| 5 | 8 | mul12s_34P | 55% | **98.24%** | 19% | **74.16%** | **23.01%** | **2.83%** |
| 6 | 8 | mul12s_2TE | 55.6% | 9.8% | 19% | 3.94% | 27.5% | 68.76% |

on the intrinsic nature of the approximated multiplier that implies to have different output variations w.r.t. the precise multiplier.

To study how the output ($c$) of a multiplier ($a \times b = c$) changes according to the possible input combinations ($a$, $b$), the following experiments were performed. Specifically, we quantified how much the output varies when the Least Significant Bits (LSBs) of $b$ change. To carry out this analysis, we fixed all the bits of $a$ and $(8-n)$ of the Most Significant Bits (MSBs) of $b$. The remaining $n$ bits of $b$ were varied from 0 to $2^n$ while storing the output of the multiplier. Finally, the mean and variance of the gathered outputs were computed. This process is repeated $2^{8+(8-n)}$ times, once for each combination of the fixed inputs, thus generating the heatmaps of figure 2. Figures 2a to 2h show the same process for different values of $n$. For instance, in Figure 2a, we vary only one bit (thus $n = 1$), in Figure 2b, we vary two LSBs (thus $n = 2$) and so on until Figure 2h in which all bits are considered ($n = 8$). Each heatmap reports the values of $a$'s bits on the $y$-axis, the values of $b$'s fixed bits on the $x$-axis, and the value of the variance is represented using colors: blue means no variation, yellow high variation. Zero-variance means that, given $a$'s and $b$'s fixed input bits, the output is always the same, regardless of the $n$ LSBs of $b$. Please note that in our experiments, **the input b corresponds to the 8-bit input of the DNN weights** (i.e. the injected inputs).

For example, figure 2c shows the variance of the outputs when varying the three LSBs of $b$. This means that each point of the heatmap is associated with 13 fixed bits (8 bits for input $a$ and 5 bits for input $b$). The remaining three are varied, the outputs gathered and variance computed, converted to a color, and plotted in the corresponding spot. Please note that NN's weights are mapped always on the input $b$ of the multiplier, which is the same input we injected in the previously explained experiments.

It can be easily noted that the multiplier `mul12s_34P` (the one of case #5) shows the smallest variation compared to the other multipliers, and equal to zero when varying up to three bits (Figures 2a, 2b and 2c). This means that faults have a smaller impact (or produce a smaller output variation) at the output of the multiplier, which in turn impacts negligibly the performance of our DNN.

## VI. Conclusion

This paper proved that AxC can significantly help to improve the resilience of a given systolic array (up to 8x reduction of critical faults) not only without any costs, but even by increasing the energy efficiency (up to 50%).

## Acknowledgments

This work has been founded by the RE-TRUSTING project, ANR-21-CE24-0015; the APROPOS project in the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956090; and the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

## References

[1] S. Ghanem, P. Kanungo, G. Panda, and P. Parwekar, "An improved and low-complexity neural network model for curved lane detection of autonomous driving system," *Soft Computing*, vol. 27, no. 1, pp. 493–504, 2023.

[2] S. Kundu, S. Banerjee, A. Raha, S. Natarajan, and K. Basu, "Toward functional safety of systolic array-based deep learning hardware accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 485–498, 2021.

[3] S. Burel, A. Evans, and L. Anghel, "Mozart+: Masking outputs with zeros for improved architectural robustness and testing of dnn accelerators," *IEEE Transactions on Device and Materials Reliability*, vol. 22, no. 2, pp. 120–128, 2022.

[4] A. Ruospo, E. Sanchez, L. M. Luza, L. Dilillo, M. Traiola, and A. Bosio, "A survey on deep learning resilience assessment methodologies," *Computer*, vol. 56, no. 2, pp. 57–66, 2023.

[5] G. S. Rodrigues, F. L. Kastensmidt, and A. Bosio, *Approximate Computing and its Impact on Accuracy, Reliability and Fault-Tolerance*. Springer International Publishing, 2022.

[6] M. Traiola, B. Deveautour, A. Bosio, P. Girard, and A. Virazel, "Test and reliability of approximate hardware," in *Approximate Computing*, pp. 233–266, Springer International Publishing, 2022.

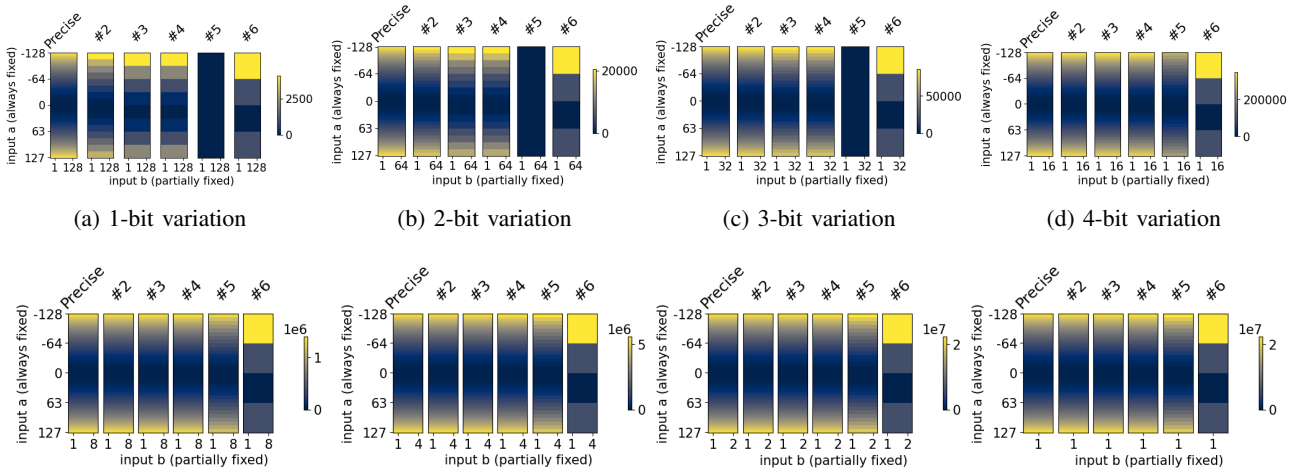
[7] M. Biasielli *et al.*, "Approximation-based fault tolerance in image processing applications," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 648–661, 2021.

[8] L. M. Luza *et al.*, "Emulating the effects of radiation-induced soft-errors for the reliability assessment of neural networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1867–1882, 2021.

[9] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[10] A. Bosio, D. Ménard, and O. Sentieys, eds., *Approximate Computing Techniques*. Springer International Publishing, 2022.

[11] S. Pappalardo, A. Ruospo, I. O'Connor, B. Deveautour, E. Sanchez, and A. Bosio, "A fault injection framework for ai hardware accelerators," in *2023 IEEE 24th Latin American Test Symposium (LATS)*, pp. 1–6, 2023.




(a) 1-bit variation (b) 2-bit variation (c) 3-bit variation (d) 4-bit variation

(e) 5-bit variation (f) 6-bit variation (g) 7-bit variation (h) 8-bit variation

Fig. 2: Outputs' variance of multipliers for a specific combination of fault-free 8-bit inputs (y-axis) and faulty 8-bit inputs (x-axis), where a growing number of bits varies.

[12] A. Ruospo, G. Gavarini, C. de Sio, J. Guerrero, L. Sterpone, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Assessing convolutional neural networks reliability through statistical fault injections," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, 2023.