# Assessing the Vulnerabilities of RISC-V using the gem5 Simulator

Mahreen Khan[1*], Maria Mushtaq[1], Renaud Pacalet[1] and Ludovic Apvrille[1]

[1]Télécom Paris, France

**Abstract**

*Emerging RISC-V processors require rigorous security evaluation to address microarchitectural vulnerabilities inherent in their rapidly evolving ecosystem. A recent paper [1] implemented both known and novel side-channel attacks targeting commercial RISC-V CPUs (U74 and C906). While this hardware-based research confirmed vulnerabilities, it could not provide detailed insights into attack dynamics. We bridge this gap using the gem5 simulation framework to systematically analyze side-channel attacks on RISC-V architectures. Our paper focuses on the access-retired attack, which exploits the unprivileged* `rdinstret` *instruction to infer protected filesystem data. By tracking retired instruction counts, attackers detect microarchitectural state differences caused by directory access checks. We utilize the gem5 simulator in full-system (FS) mode to capture kernel-level behaviors, allowing us to analyze critical performance metrics including instruction retirement, cache performance, and branch prediction statistics. This detailed simulation-based analysis is essential for understanding the behavior of the attack and for developing effective countermeasures. Advancing RISC-V security research with simulation tools like gem5 is thus a promising direction for mitigating future side-channel vulnerabilities.*

## Introduction

The open-source RISC-V instruction set architecture (ISA) has emerged as a transformative force in computing, offering unprecedented flexibility for custom hardware design. However, its rapid adoption in safety-critical domains, from embedded systems to data centers, demands rigorous scrutiny of microarchitectural security. While RISC-V's modularity enables performance and power optimizations, it also introduces attack surfaces absent in traditional ISAs. Recent work by Gerlach et al. [1] demonstrated this risk empirically, implementing novel side-channel attacks on commercial RISC-V cores (U74, C906) that exploit microarchitectural resource contention. Their hardware-based validation confirmed vulnerabilities but left critical gaps in understanding attack mechanics, particularly how ISA-specific features interact with microarchitectural states.

To address this limitation, we propose simulation-driven analysis using the gem5 framework [2], which provides cycle-accurate modeling of RISC-V processors in full-system (FS) mode. We focus on the `rdinstret`-based access-retired attack, where adversaries infer filesystem metadata through retired instruction count variations. Unlike hardware experiments, gem5 enables granular tracing of kernel-level events, cache states, and pipeline behaviors during attack execution. This methodology reveals how directory access checks, ostensibly protected by privilege boundaries, create measurable timing differences through cache line contention and branch mispredictions.

*Corresponding author: `mahreen.khan@telecom-paris.fr`

## gem5 Methodology

To analyze the access-retired attack on RISC-V, we configure gem5 with the following setup:

1. **Simulation Mode**: We use **Full-System (FS) mode**, which models the complete hardware-software stack, including the operating system. This is essential for capturing kernel-level interactions during filesystem access checks.
2. **Disk Image and Kernel**: We configure gem5 for RISC-V, our target architecture, and modify the RISC-V disk image to include the attack binary. The system boots using the `riscv-bootloader-vmlinux-5.10` kernel.
3. **CPU Model**: Use the `O3CPU` out-of-order processor model to simulate speculative execution and branch prediction behaviors. This enables tracking of pipeline stalls and branch mispredictions.
4. **Memory System**: Configure cache hierarchies (L1/L2) and memory system [2].

Figure 1 summarizes the workflow for simulating gem5 in full-system mode.

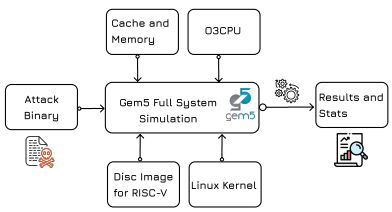



**Figure 1:** *Gem5 full-system simulation workflow for analyzing the access-retired attack on RISC-V.*

# Results

The gem5 simulator successfully implemented and analyzed the access-retired attack on the RISC-V architecture. The results align closely with hardware measurements, demonstrating gem5's ability to model detailed microarchitectural behavior and provide insights into attack mechanisms.

Our experiments show a clear side-channel signal through systematic differences in microarchitectural metrics when accessing existing versus non-existing files. As shown in Figure 2, the retired instruction count increases by 25% (from 3,300 to 4,120) when a target file exists, even though the system calls return NULL in both cases. This increase is due to extra kernel checks, such as permission validation.

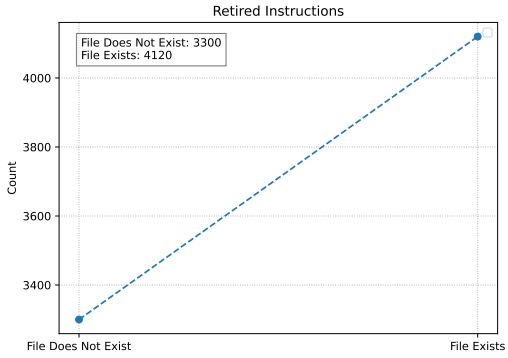



**Figure 2:** *Retired instruction counts comparison.*

The simulated execution time nearly doubles (from 65.61s to 151.30s) when the file exists, as illustrated in Figure 3. This extended duration correlates with the higher retired instruction count.

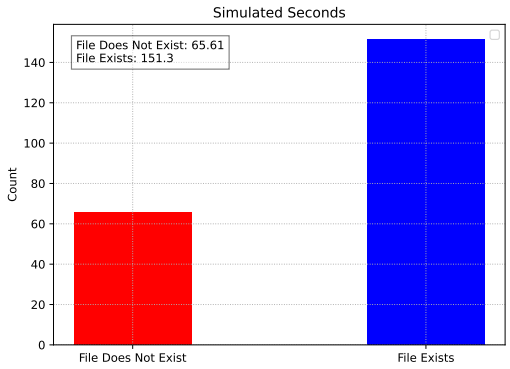



**Figure 3:** *Simulated execution time comparison.*

Additionally, the total number of simulated instructions rises by 20% (from 267M to 320M) for file existence as shown in Figure 4. Memory subsystem behavior also differentiates file existence, as evidenced by increases in branch fetching, indirect branch lookups, and reorder buffer entries. Table 1 summarizes the metrics comparing file existence and non-existence scenarios. These consistent differences confirm that failed system calls leak file existence information through multiple microarchitectural vectors.

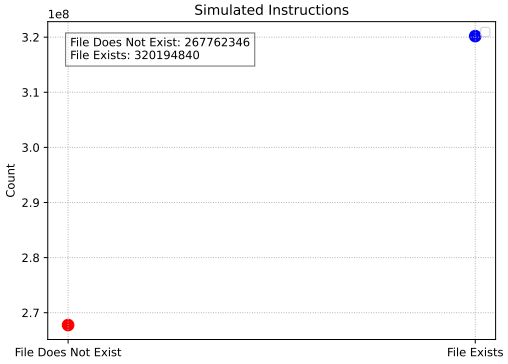



**Figure 4:** *Total instruction counts comparison.*

**Table 1:** *Comparison of Metrics for File Existence*

| Metric | File Exists | File Does Not Exist |
|---|---|---|
| Retired Instructions | 4,120 | 3,300 |
| Indirect Branch Lookups | 3,915,281 | 2,825,809 |
| Load Instructions Executed | 71,511,594 | 55,635,190 |
| Branches Fetched | 90,168,771 | 76,010,909 |
| Reorder Buffer Total | 678,320,302 | 575,317,912 |
| Simulated Seconds | 151.30 | 65.61 |
| Simulated Instructions | 320,194,840 | 267,762,346 |

# Future Work

Future work could focus on developing a security research platform for RISC-V based on gem5, providing a modular framework for analyzing vulnerabilities. This platform could include customizable templates for cache and pipeline designs, and automated tools for profiling and visualizing key metrics like speculative execution and cache behaviors. Integrating standardized APIs would further enhance its versatility, enabling researchers to explore attack vectors and mitigation strategies more effectively.

# Conclusion

This paper demonstrates the feasibility of using gem5 for RISC-V security research by implementing and validating access-retired attack. Simulation-based approaches provide a powerful tool for advancing RISC-V security, enabling researchers to identify and mitigate vulnerabilities in a cost-effective and scalable manner.

# References

[1] Lukas Gerlach et al. "A security RISC: microarchitectural attacks on hardware RISC-V CPUs". In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 2321–2338.

[2] Jason Lowe-Power. *Gem5 Documentation*. [Online]. Available: `https://www.gem5.org/documentation/`. Sat 29 June 2024.

# STDP-Trained Spiking Neural Network Reliability Assessment Through Fault Injections

Zalfa Jouni and Haralampos-G. Stratigopoulos
Sorbonne Université, CNRS, LIP6, Paris, France

## I. Introduction

Spiking Neural Networks (SNNs) represent a promising computing paradigm for hardware-based AI that closely mimics the human brain mechanism [1]. In contrast to conventional Artificial Neural Networks (ANNs), SNNs deliver low power consumption and rapid inference, making them especially suited for edge computing [1]. Main training approaches include gradient-based backpropagation adaptations and ANN-to-SNN conversion. In addition, spike-timing dependent plasticity (STDP) stands as a well-known, biologically plausible rule. Prior studies show that STDP proves effective for on-chip unsupervised learning. For example, Intel's Loihi neuromorphic processor has on-chip learning rules that allow local, unsupervised learning using STDP-based synaptic updates [2].

The design of efficient SNN hardware accelerators remains an intense research field. Many assume that SNNs inherit the fault tolerance of the human brain. However, recent fault injection studies enabled by automated fault injection frameworks challenge this assumption [3].

The vast majority of previous fault injection studies in SNNs considered traditional convolutional and fully-connected SNNs trained with gradient-based backpropagation [4]. In this work, we study the reliability of STDP-trained SNNs under hardware faults which is largely unexplored.

## II. Fault Injection Setup

### A. SNN case study and training framework

The fault injection experiment is conducted on the STDP-trained SNN architecture shown in Fig. 1 [5] using the MNIST dataset. Pixel intensities are converted into Poisson-distributed spike trains and fed into the network's first layer. The first layer has 784 input neurons (one per pixel), and these neurons are fully-connected, via excitatory synapses, to a second layer of 400 excitatory neurons. Each excitatory neuron in the second layer is paired one-to-one with an inhibitory neuron via excitatory synapses. The neuron follows the leaky integrate-and-fire (LIF) model, chosen for its biomimetic and simple behavior. For the purpose of the experiment, we utilize the open-source, Python-based Brian 2 simulator for SNNs [6], leveraging GPU acceleration.

This work was funded by the ANR RE-TRUSTING project under Grant N$^{o}$ ANR-21-CE24-0015-03 and by the European Network of Excellence dAIEDGE under Grant Agreement N$^{o}$ 101120726.

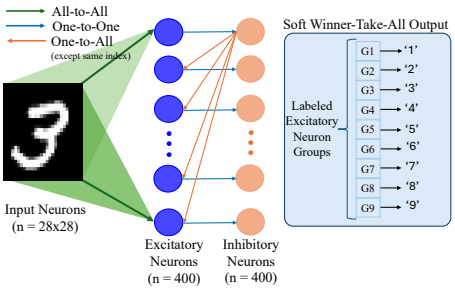



Fig. 1: SNN architecture for STDP learning [5].

### B. Fault model

We consider three fault models : (1) Dead neurons, which halt their spiking activity regardless the input; (2) Saturated neurons, which continuously fire regardless the input; and (3) Synapse weight perturbations, modeled as bit flips considering that the weights are stored as digital words in on-chip memory.

### C. Fault injection scenarios

We consider the following fault injection scenarios grouped under three categories.

*1) Faults occurring after training:* STDP learning is conducted on a fault-free network, but permanent faults may arise over the hardware's lifetime, potentially impacting inference. This scenario is analogous to training the SNN in software and then deploying it onto faulty hardware. We consider three scenarios:

*a) Scenario 1a:* Single and multiple dead neurons.

*b) Scenario 1b:* Single saturated neurons. We do not consider multiple saturated faults since, as we will see, even a single saturated neuron fault drastically affects the classification accuracy.

*c) Scenario 1c:* Synapse weight perturbation faults considering different bit error rate (BER) probabilities.

The objective is to categorize different fault types and locations as either benign or critical, and study the classification drop as a function of fault density for neurons and BER for synapses.

*2) Faults occurring prior to training:* STDP learning is conducted on-chip on a faulty network, which may impact training quality. These faults persist during inference. We consider two scenarios:

*a) Scenario 2a:* Single and multiple dead neurons.

*b) Scenario 2b:* Single saturated neuron.

The goal here is to assess whether biologically inspired STDP learning can adapt to faulty hardware, demonstrating

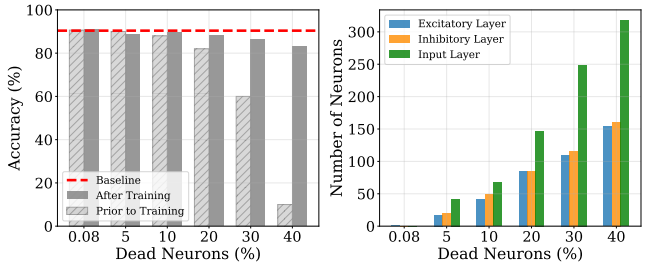



Fig. 2: Accuracy as a function of percentage of randomly distributed dead neurons introduced before and after the training process.

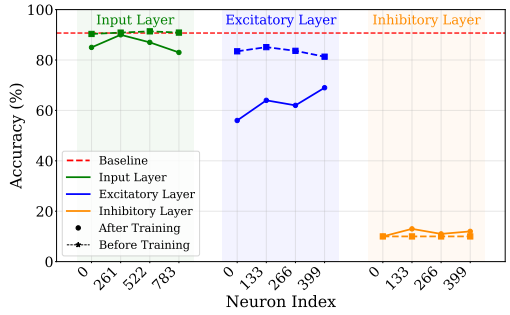



Fig. 3: Impact of a single saturated neuron occurring before and after the training process. The x-axis shows the neuron index in the input (green), excitatory (blue) and inhibitory (orange) layers.

inherent robustness. Note that synapse faults in this category are irrelevant since they are overwritten during training.

*3) Transient faults occurring during training:* Here, we assume that learning takes place on-chip and radiation, voltage and temperature fluctuations, etc., affect synapse weights during the process.

*a) Scenario 3a:* The memory storing the synapses undergoes a BER at a random instant of learning process.

The aim is to determine whether learning can still succeed despite transient faults. Bit-flips are regarded as transient since weights may be updated throughout the training process.

## III. Fault Injection Results

*1) Faults occurring after training:*

*a) Scenario 1a:* The results illustrated in Fig. 2 show that classification accuracy remains largely stable around the baseline of 90.7%, even with dead neuron fault densities as high as 10%. Additionally, neither the proportion of dead neurons within a specific layer nor their exact locations within the layer seem to play a role.

*b) Scenario 1b:* As shown in Fig. 3, the accuracy drop varies with the neuron location within the layer, but as a general observation it is significant for all neurons and appears to be more pronounced for the inhibitory layer. This is because a saturated neuron in the inhibitory layer suppresses the activity of all neurons in the excitatory layer except the one that connects to it, thus the same class is permanently selected.

*c) Scenario 1c:* For each BER probability, 100 experiments were conducted, and the average, minimum, and maximum accuracy are reported in Fig. 4. When the BER is below $10^{-2}$, a relatively high and likely unrealistic value, the accuracy remains close to the baseline.

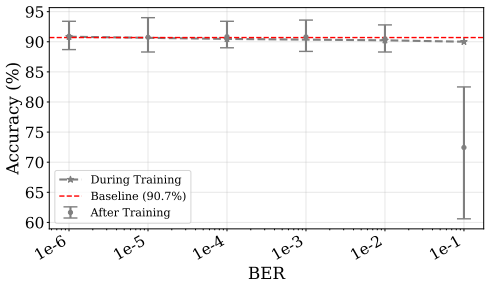



Fig. 4: Accuracy as a function of BER probability in synaptic weights, introduced during and after training.

*2) Faults occurring prior to training:*

*a) Scenario 2a:* The results illustrated in Fig. 2 show that the network maintains a relatively high accuracy, experiencing only a minor accuracy drop of less than 5%, when up to 10% of the neurons are dead.

*b) Scenario 2b:* As shown in Fig. 3, saturation in the input layer has no effect on classification accuracy. In the excitatory layer, accuracy can decline by up to 10%, whereas in the inhibitory layer, it is completely compromised reducing from the baseline value by 83%.

*3) Transient faults occurring during training:*

*a) Scenario 3a:* Fig. 4 shows that even for an extremely high BER of $10^{-1}$ occurred during training, the learning process successfully completes, achieving an accuracy very close to the baseline. This experiment demonstrates that the network training can adapt to a high rate of multiple bit-flips during the training process.

## IV. Conclusion

Our experiments demonstrated that dead neurons up to 5% or synaptic weights affected by a BER as high as $10^{-2}$ do not impact classification accuracy, and learning can compensate for them. In contrast, learning fails when saturated neurons are present in the excitatory and inhibitory layers, and accuracy is significantly affected if neurons become saturated during the chip's lifetime, with the last inhibitory layer being the most critical. These findings suggest that testing and fault tolerance strategies could focus exclusively on saturated neuron faults, thereby significantly reducing costs.

## References

[1] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nat. Comput. Sci.*, vol. 2, no. 1, pp. 10–19, Jan. 2022.

[2] G. Orchard *et al.*, "Efficient neuromorphic signal processing with Loihi 2," in *IEEE Workshop Signal Process. Syst. (SiPS)*, Nov. 2021, pp. 254–259.

[3] T. Spyrou, S. Hamdioui, and H.-G. Stratigopoulos, "SpikeFI: A fault injection framework for spiking neural networks," *arXiv:2412.06795*, 2024.

[4] H. Stratigopoulos, T. Spyrou, and S. Raptis, "Testing and reliability of spiking neural networks: A review of the state-of-the-art," in *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Nanotechnol. Syst. (DFT)*, Oct. 2023.

[5] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, Aug. 2015.

[6] M. Stimberg, R. Brette, and D. FM Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, Oct. 2019.

# Evaluation of the Interactions Between Cybersecurity and Reliability Mitigations in the Internet of Things

Théo Serru, Guillaume Andrieux, Olivier Pasquier, Sébastien Pillement
*Nantes Université, CNRS,*
*IETR-UMR 6164*
F-44000, Nantes, France
{theo.serru, guillaume.andrieux, olivier.pasquier, sebastien.pillement}@univ-nantes.fr

***Abstract*—Environmental conditions, low power, high connectivity and heterogeneous nature of internet of things (IoT) make it very vulnerable to cyberattacks and failures. In addition, security and reliability can influence each other, especially, countermeasures can increase one notion at the cost of the other. With the development of IoT in critical fields such as energy, transportation or healthcare, safety has become a bigger concern. It is therefore paramount to evaluate, not only security and reliability separately, but also to measure their interactions.**

**In this work, we propose to investigate the challenge of assessing security measures' reliability, and reliability measures' security. Our goal is to develop a methodology that gives quantitative indicators on the modifications of security and reliability induced by a countermeasure. With this evaluation, architects will be able to compare several implementations with security and reliability in mind.**

***Index Terms*—Cybersecurity, Reliability, IoT, 5G.**

## I. Introduction

Internet of Things (IoT) is built upon connected objects and high connectivity networks such as 5G. These devices are integrated in the environment for monitoring, data processing and decision making. IoT specificities like harsh environmental conditions, their low-power nature, or the possibility to harvest energy may negatively impact the reliability. Cybersecurity is also a big concern in IoT due to its highly connected nature. For this reason, many works have been studying cyberattacks, failures and their impacts at every layer of the IoT architecture.

In addition, failures and cyberattacks interact with each other. For instance, a failure in the power supply might help the attacker to penetrate a system. Conversely, the famous Stuxnet attack showed that a coordinated attack can lead to physical failures with catastrophic consequences.

These interactions imply that reliability, safety and cybersecurity can no longer be considered as separate fields, especially when dealing with cyber-physical systems.

In the literature, researchers working on reliability, safety or security modeling proposed works combining these notions [1], [2], [3]. Their goal is to model the behavior of a system in the case of security or reliability events and to modify the architecture with countermeasures. With a formal modeling, one can build safe-and-secure-by-design architectures. However, these works lack of validation [1], especially on the real impacts that the integration of countermeasures will have on the system (and at different levels of abstraction).

This work is funded by the French National Research Agency under the France 2030 label (NF-HiSec ANR-22-PEFT-0009).

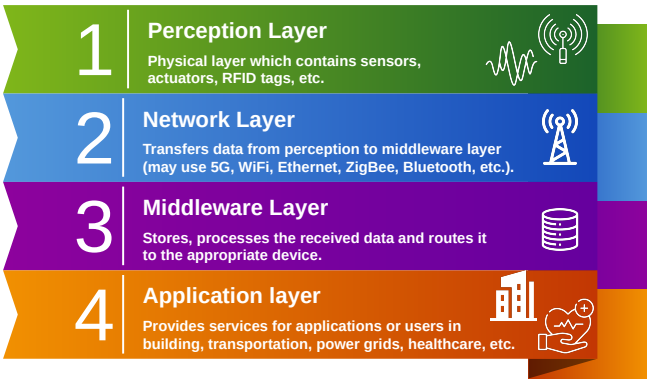



Fig. 1. The Four Layer IoT Architecture.

In this work, we propose to investigate how security and reliability countermeasures might impact reliability and security respectively. We want to highlight the existing interactions (reinforcement, conditional dependency, conflict or independence) and quantify them with appropriate metrics. To this end, we will focus on countermeasures at the perception layer and develop a method for the evaluation of reliability and security of countermeasures.

## II. IoT

### A. Architecture

IoT is composed of sensors, actuators, gateways, cloud services, networks and application servers. Traditionally, the architecture of IoT is divided in several layers, with no real agreement on the number of layers due to the nonuniformity and lack of standardization [4].

Here, we give a quick overview of the four-layer architecture.

*Perception layer:* corresponds to the physical layer where sensors collect data about the environment. Sensor nodes are composed of sensors, a transceiver, a processor and a power unit. They mostly communicate wirelessly (in a wireless sensor network or WSN) to pass the information to the upper layers.

*Network layer:* carries and transfers data from perception layer to servers. The unprocessed data can be filtered and preprocessed for analysis.

*Middleware layer:* have large storage and processing capabilities. It processes the received data and routes it to the appropriate device.

*Application layer:* is the interface that provides services for the application or user.

The sensor layer is cyber-physical by nature, embedding sensors, actuators and communication capabilities in a potentially harsh environment. Because of the environment and the resource constraints, this layer is more prone to failures which have a negative impact on the overall safety of IoT [3]. In addition, many countermeasures in this layer are physically implemented, which gives the opportunity to evaluate the reliability and security of hardware and software. Therefore, we will start our work by studying this layer.

### B. Reliability and Security Events in IoT

Numerous reliability and security events can occur in sensor nodes and WSN. Table I displays attacks (from [4], [5]) and reliability faults (from [6]) that may lead to failures of the perception layer, and some of their countermeasures.

TABLE I
CYBERATTACKS, FAULTS AND COUNTERMEASURES FOR THE PERCEPTION LAYER

| | **Security** | **Reliability** |
|---|---|---|
| Attacks/Faults | Malicious node attack | Low energy |
| | False injection attack | Miscalibration |
| | Hardware attacks | Bit-flip |
| | Unauthorized admittance | Limited memory fault |
| | Side channel attack (SCA) | Memory errors |
| | Eavesdropping/Sniffing | High battery temperature |
| | Node Cloning | Battery low voltage |
| | Battery drainage attack | Interference |
| | Integrity attacks | Broken connector |
| | Replay attack | Material decay |
| | | Environmental effects |
| | | Data processing faults |
| | | Software design faults |
| | | Incorrect algorithms |
| Countermeasures | Detection | Redundant design |
| | Physical layer security | Hardware redundancy |
| | Targeted defense | Software redundancy |
| | Physical layer security | Information redundancy |
| | Access control | Data redundancy |
| | Algorithm-based | Time redundancy |
| | Channel management | Fault tolerance |
| | Encryption | |
| | Authentication | |
| | Cryptography | |

In Table I, one might identify interactions between reliability and security countermeasures. For instance, adding a detection module will lower the reliability by adding new components that can fail. It will however increase the security and reliability by detecting faults and allowing an intervention before any accident happens.

## III. PERSPECTIVES

The first step of our work is a thorough analysis of attacks, failures and their countermeasures. With this analysis, we want to highlight the links between faults, countermeasures or evaluation means (e.g. metrics) used in reliability and security.

The second step is to develop an approach to evaluate the reliability of a cybersecurity measure, and conversely. This methodology will consider the specificities of IoT and the previous step to tailor our analysis to the countermeasure evaluated. If the approach intends to be usable for every countermeasure, it will pay special attention to the parameters highlighted at step one. As we are dealing with the perception layer, we will consider both hardware and software countermeasures and impacts.

Finally, with enough confidence in the analysis at the node level, we will evaluate the impact of these countermeasures at the (local) network level. At this level, we might identify new interactions, or known consequences might have different impacts if several nodes embed the countermeasure.

## IV. CONCLUSION

Cybersecurity and reliability are major concerns for the internet of things. In addition to traditional threats and failures, implementing countermeasures might increase security at the cost of reliability (or conversely). There is thus an urgent need for holistic methods that evaluate the impact of countermeasures on both reliability and security. This work will benefit the evaluation of countermeasures by considering performance and potential side-effects. Numerical indications on the modifications of reliability and/or security induced by a countermeasure may then be implemented in model-based analyses for a system or network level analysis.

## REFERENCES


[1] G. Kavallieratos, S. Katsikas, and V. Gkioulos, "Cybersecurity and Safety Co-Engineering of Cyberphysical Systems—A Comprehensive Survey," *MPDI Future Internet*, vol. 12, no. 4, 2020.

[2] T. Serru, N. Nguyen, M. Batteux, and A. Rauzy, "Minimal critical sequences in model-based safety and security analyses: Commonalities and differences," *ACM Trans. Cyber-Phys. Syst.*, vol. 7, no. 3, Jul. 2023. [Online]. Available: https://doi.org/10.1145/3593811

[3] A. Abdulhamid, S. Kabir, I. Ghafir, and C. Lei, "An Overview of Safety and Security Analysis Frameworks for the Internet of Things," *Electronics*, vol. 12, no. 14, p. 3086, Jul. 2023.

[4] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, Oct. 2020.

[5] W. Fei, H. Ohno, and S. Sampalli, "A Systematic Review of IoT Security: Research Potential, Challenges, and Future Directions," *ACM Comput. Surv.*, vol. 56, no. 5, pp. 111:1–111:40, Nov. 2023.

[6] M. Melo and G. Aquino, "The Pathology of Failures in IoT Systems," in *Computational Science and Its Applications – ICCSA 2021*, O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, and C. M. Torre, Eds. Cham: Springer International Publishing, 2021, pp. 437–452.

# Atténuation à faible coût des chevaux de Troie matériels dans un NoC utilisant une compression basée sur la technique Delta

Hamza Amara*, Cédric Killian §, Daniel Chillet* and Emmanuel Casseau*

* Univ. Rennes, Inria, CNRS, IRISA, § Univ. Jean-Monnet Saint-Etienne, Lab. hubet Curien, CNRS

***Abstract*—Les systèmes multiprocesseurs sur puce (MPSoC) et les réseaux sur puce (NoC) sont des composants clés des architectures modernes, permettant une communication efficace entre les unités de traitement, mais peuvent souffrir d'une latence importante et d'une forte consommation de bande passante. La compression Delta atténue ce problème en exploitant la redondance des données a transmettre, mais augmente la vulnérabilité aux erreurs et aux attaques, notamment celles des chevaux de Troie matériels (HT). Cet article présente DyED2C, une technique de protection à faible coût qui adapte dynamiquement les codes de détection et de correction d'erreurs (EDC/ECC) pour sécuriser les métadonnées des paquets compressés par une compression basée sur la technique Delta, en utilisant les bits libres des flits d'en-tête. Les résultats expérimentaux sur des données de Cifar-10 transitant dans un NoC montrent que DyED2C réduit l'erreur quadratique moyenne (MSE) de 43% par rapport au code de Hamming.**

***Index Terms*—Réseau sur puce, Compression basée sur la technique delta, Sécurité du NoC, Cheval de Troie matériel**

## I. Introduction

Le développement des systèmes sur puce multiprocesseurs (MPSoC) a révolutionné l'informatique, permettant des architectures avancées et flexibles. Le réseau sur puce (NoC) [1], élément clé des MPSoC, assure la communication. Toutefois, les composants du NoC proviennent généralement de propriétés intellectuelles (IP) tierces et peuvent contenir des cheveaux de Troie matériels (HT). L'augmentation du nombre de cœurs et des échanges intensifs accroît la latence et la consommation de bande passante. Pour atténuer ce problème, des techniques de compression légère nommées Delta, comme FlitZip [2] et Impv_Delta [3], exploitent la redondance des données pour réduire la taille des paquets. Toutefois, ces méthodes reposent sur des bases communes pour reconstruire les données, les rendant vulnérables aux modifications. Les HT dans les routeurs du NoC peuvent injecter des fautes dans les paquets compressés, compromettant ainsi l'intégrité des données. Ce travail propose une méthode d'atténuation à faible coût, intégrant dynamiquement des codes EDC/ECC dans les flits d'en-tête pour protéger les bases, offrant ainsi une protection efficace contre les attaques HT.

## II. TRAVAUX RELATIFS

### A. Compression de paquets basée sur delta

La compression delta repose sur une base commune $B$ pour les données $\{D_i\}$ calculant les différences $\{\Delta_i\}$ entre elles. La taille du champ alloué pour la base dans le flit d'en-tête est égale à la taille des données $S_D$. FlitZip [2] compresse les données utiles du flit en utilisant une taille de base ($S_B$ = $S_D$), tandis que Impv_Delta [3] simplifie cette compression en stockant les bits MSB communs dans la base et en transmettant les bits LSB non similaires dans le flit de données utiles compressé. Cela permet de réduire la taille de la base ($S_B \leq S_D$). Notre travail exploite cette caractéristique pour inclure des techniques de protection dans les bits libérés.

### B. HT au sein des NoC

La protection des NoC contre les HT a été largement étudiée [4], notamment les attaques de déni de service (DoS) via des routeurs compromis falsifiant les paquets.

Kulkarni *et al.* [5] ont introduit un HT modifiant les adresses de destination des paquets, entraînant des DoS sans proposer de contre-mesures. Dans [6], les auteurs ont étendu cette attaque aux champs critiques d'en-tête, dégradant les performances. Leur solution, basée sur le mélange de bits et le code Hamming, impose un surcoût matériel sans empêcher totalement l'activation des HT.

Cependant, ces travaux ignorent l'impact des HT sur les paquets compressés. La compression Delta améliore l'efficacité du NoC mais fragilise l'intégrité des bases, exposées aux fautes injectées. Or, la protection des bases par des méthodes classiques (mélange de bits, EDC/ECC) est limitée par le faible nombre de bits libres dans le flit d'en-tête. Même Impv_Delta, qui réduit la taille des bases, complique l'intégration d'une protection efficace, nécessitant une adaptation aux bits librés.

Pour y remédier, nous proposons une technique d'atténuation à faible coût ajustant dynamiquement l'usage des bits libérés par la compression pour protéger les bases.

## III. Détail de l'attaque

Nous supposons la présence d'un cheval de Troie matériel (HT) dans un routeur malveillant ($R^{HT}$), capable de corrompre les données des paquets pendant leur transmission. L'attaque consiste à inverser la valeur des bits de base dans les flits d'en-tête et/ou des données utiles des flits.

Dans **Fig. 1**, nous illustrons une attaque HT dans un NoC utilisant une compression basée sur la technique delta. Les données d'images brutes sont mises en paquets et compressées au niveau de l'interface réseau source $NI_{12}$, puis acheminées via le routeur malveillant $R^{HT}_{14}$. Lorsque les paquets passent par $R^{HT}_{14}$, le HT altère les paquets victimes. À l'arrivée à l'interface réseau de destination $NI_{10}$, les paquets sont décompressés et les données reçues au niveau de $IP_{10}$ sont modifiées, ce qui dégrade l'application qui les utilise.

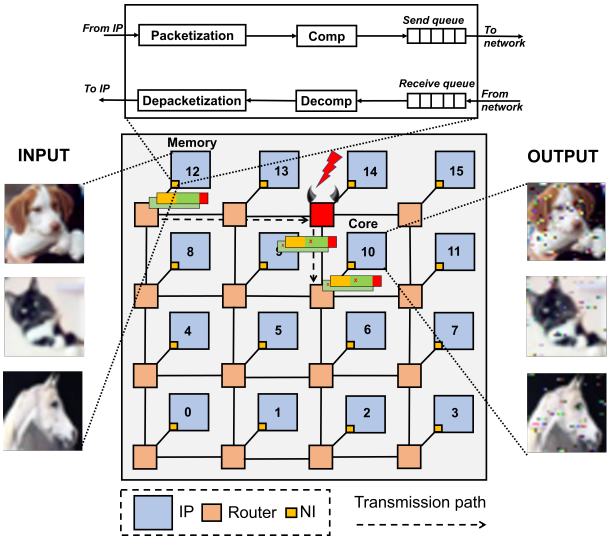



Fig. 1: Scénario d'attaques HT dans un NoC utilisant une technique de compression.

## IV. Technique DyED2C proposée

### A. Principe

DyED2C est une méthode dynamique de détection et correction d'erreurs conçu pour protéger les bases de paquets compressés contre les attaques basées sur l'injection des fautes. Contrairement aux méthodes statiques, DyED2C exploite les bits libérés dans les bases de taille variable pour assurer la protection sans dépasser le champ alloué $S_D$. **Fig. 2** illustre comment DyED2C utilise les bits libérés dans les bases de taille variable pour intégrer la protection, sans dépasser le champ alloué.

### B. Implémentation de DyED2C

*1) Architecture matérielle:* Le module DyED2C étend les interfaces réseaux (NI) avec deux blocs matériels : DyED2C$_C$ (après compression) et DyED2C$_D$ (avant décompression). Ces deux blocs partagent une architecture commune, DyED2C$_D$ intégrant un bloc supplémentaire de correction des fautes.

L'architecture optimise le coût matériel en utilisant un seul code de Hamming, ajustable dynamiquement aux bases variables. Les codes de Hamming sont choisis pour leur efficacité à corriger les erreurs avec un surcoût en bits minimal. Bien que des codes plus puissants comme Reed-Solomon ou BCH existent, ils restent plus coûteux en ressources.

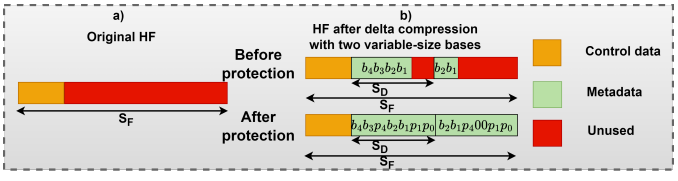



Fig. 2: Exemple d'intégration de la protection à l'aide de bits libres dans des bases de taille variable.

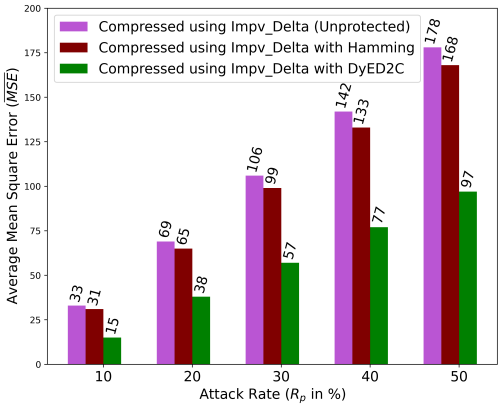



Fig. 3: Évaluation de la réduction de $\overline{MSE}$ sur différents taux d'attaque après utilisation de la protection.

### C. Évaluation des performances

Nous évaluons l'efficacité de DyED2C avec Impv_Delta [3] en utilisant la métrique d'erreur quadratique moyenne (MSE) sur Cifar-10[1]. L'analyse inclut un scénario sans protection et un code Hamming étendu (limité à 4 bits), en variant le taux d'attaque $R_P$ (10%-50%) et le nombre de fautes $n_f$ (1-5), avec $S_F$ = 32 bits et $S_D$ = 8 bits. Le HT cible les bases et les données utiles des flits. **Fig. 3** montre que $\overline{MSE}$ augmente avec $R_P$, mais DyED2C réduit ce $\overline{MSE}$ de 46% par rapport à l'absence de protection et de 43% face au Hamming étendu. Cette amélioration vient de sa protection dynamique sur différentes tailles de base, contrairement à la limitation à 4 bits du Hamming étendu.

## V. Conclusion

Dans cet article, nous avons présenté DyED2C, une technique de protection dynamique et peu complexe exploitant les bits disponibles des flits d'en-tête des paquets traversant le NoC. Comparée au code Hamming, DyED2C atténue efficacement les attaques par injection de fautes, réduisant le MSE jusqu'à 43% sur Cifar-10 avec la compression Impv_Delta.

## References

[1] M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, J. Balkind, A. Lavrov, M. Shahrad, S. Payne, and D. Wentzlaff, "Piton: A Manycore Processor for Multitenant Clouds," *IEEE Micro*, Mar. 2017.

[2] D. Deb, R. M.K, and J. Jose, "FlitZip: Effective Packet Compression for NoC in MultiProcessor System-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, Jan. 2022.

[3] N. C A, D. M. Viswanathan, and R. K. James, "An Improved Delta Compression Technique for NoC Packets," in *2023 International Conference on Next Generation Electronics (NEleX)*.

[4] S. Charles and P. Mishra, "A Survey of Network-on-Chip Security Attacks and Countermeasures," *ACM Computing Surveys*, Jun. 2022.

[5] V. J. Kulkarni, R. Manju, R. Gupta, J. Jose, and S. Nandi, "Packet header attack by hardware trojan in NoC based TCMP and its impact analysis," in *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*. Virtual Event: ACM, Oct. 2021.

[6] M. K. J.Y.V., A. K. Swain, S. Kumar, S. R. Sahoo, and K. Mahapatra, "Run Time Mitigation of Performance Degradation Hardware Trojan Attacks in Network on Chip," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2018.

[1]Cifar-10 contient 60 000 images 32x32 avec des pixels de 8 bits.

# Distributed Machine Learning Inference Using Commodity SoC-FPGAs: First Results

Mathieu Hannoun*†
*MADICOB
†Laboratoire ETIS,
UMR 8051,
CY Cergy Paris Universités,
ENSEA, CNRS,
F-95000, Cergy, France
ORCID: 0009-0005-0449-2668

Stéphane Zuckerman
Laboratoire ETIS,
UMR 8051,
CY Cergy Paris Universités,
ENSEA, CNRS,
F-95000, Cergy, France
ORCID: 0000-0002-8586-0477

Olivier Romain
Laboratoire ETIS,
UMR 8051,
CY Cergy Paris Universités,
ENSEA, CNRS,
F-95000, Cergy, France
ORCID: 0000-0002-2172-1865

***Abstract*—Deep Neural Networks (DNNs) have experienced significant growth over the years, accompanied by a corresponding rise in energy consumption due to their escalating demand for computational resources. To mitigate the environmental impact of AI and address growing concerns over data privacy, a growing trend is to process data locally at the edge, rather than relying on large-scale data centers. FPGA-based systems are particularly suited for this kind of applications, with their low power consumption to high parallel computation ratio. The main drawback of commodity FPGAs is their limited hardware resources, impacting how big DNNs can be to run efficiently on such targets. We present a methodology for distributed DNNs on multiple commodity FPGAs to support models that are usually only suited for larger FPGAs. We are able to support the inference for a MobileNetV1 on six Zedboards.**

***Index Terms*—Machine Learning, Deep Learning, CNN, FPGA, Dynamic Reconfiguration, Edge Computing**

## I. Introduction

FPGAs struggle with large-scale DNNs or CNNs due to the high computational demands of floating-point operations. However, frameworks such as FINN [1] incorporate weight compression and quantization techniques (*e.g.*, as few as one or two bits for weights and biases [2]), alongside operation conversions [3], [4]. While works on high-end devices (*e.g.*, AMD Alveo) have shown promise for DNN inference [5], their usage remains limited to data centers due to cost and energy consumption. In contrast, commodity FPGAs offer an appealing compromise for edge computing because of their lower power footprint and ability to exploit parallelism, though their limited resources constrain them to smaller networks.

The goal of this work is to demonstrate the feasibility of splitting larger models, which can only fully fit on high-end FPGAs (*i.e.*, UltraScale+, *etc.*), on a cluster of commodity FPGAs, using the FINN framework. This eventually could lead to a low-power distributed edge solution capable to adapt to a wide range of DNNs and tasks. We propose a way to distribute DNN models inference over several FPGAs when a single board is not sufficient to hold the whole DNN. We leverage dynamic reconfiguration of a SoC-FPGA system.

The problem we address in this work is how to fit DNNs into embedded devices while leveraging FPGAs suitability for hardware acceleration and their low power consumption. In addition, we are interested in the possibility to substitute a dedicated high-end FPGA for a multiple of networked low-power commodity FPGAs.

## II. Methodology

Our approach enables the deployment of DNNs, typically suited for high-end FPGAs (*e.g.*, Alveo U250) due to their size, onto commodity FPGAs by partitioning the model into multiple sub-models. These sub-models are distributed across a network of SoC-FPGAs. Our methodology provides a flexible and scalable approach to FPGA-based inference, enabling larger models to run on low-cost hardware. While it does not yet match the performance of high-end FPGAs, it represents a step toward more accessible and efficient distributed inference solutions. Our work uses FINN as a basis for hardware implementation to run inferences. FINN can convert a DNN model to target an FPGA. It will take an ONNX model representing the DNN architecture and its weights, convert it to multiple intermediate representations and output a bitstream including the model and custom Direct Memory Access (DMA) engines. However, the intended use is to take a complete model to produce a configuration that will fit in one single FPGA. Our goal is to partition the model into smaller sub-models that can each fit into a commodity FPGA and communicate with each other to complete the overall model. To achieve this, we split the ONNX model representing the DNN, use FINN to generate a bitstream from each sub-model, use those bitstreams to configure each board, and establish communication with every board to run inferences.

### A. Network splitting

To allow medium size DNN models to fit on low-power FPGAs, it is necessary to split them to fit resource constraints. FINN [4] is used to generate a preprocessed model and an initial resource estimation. The resulting split is based on those models. The objective of our automated splitter is to maximize resource occupancy to reduce the number of bitstreams, since dynamic reconfiguration and network operations are the most

costly in time (detailed in Section III-B). To find a suitable cut, the ONNX graph is traversed, starting from the first layer. For each node, the required resources estimate (*i.e.*, LUTs, B-RAMs, DSPs) are added, until they exceed available resources for a given board. ONNX manipulations are then used to split the preprocessed model. The resulting models are then fed to FINN to generate FPGA design as well as a bitstream for each sub-model. We modified FINN to support the Zedboard for bitstream generation (since we do not use the Pynq OS, we have no need for the Pynq overlay).

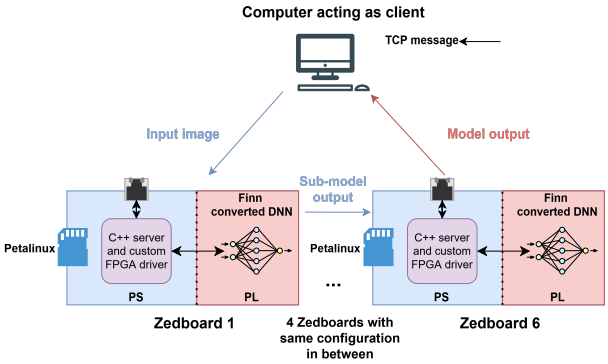



Fig. 1. Example of a full communication for a DNN divided into six bitstreams, from an image input to the model classification (Sub model 6 output), from right to left in a sequential manner.

### B. System design

Each SoC-FPGA board runs its own client/server, the client sends data to the next board while the server waits and processes data from the previous board (see Fig. 1). Sub-models are generated beforehand (see subsection II-A) and each board has all sub-models stored locally. Communications are handled by a TCP client/server written in C++, and running on the Processing System (PS) side, as shown in Fig. 1. The server directly interacts with the Programmable Logic (PL) part to trigger network inferences through the ARM CPU. DNNs are fully executed on the FPGA fabric, with no contribution from the CPU.

The client sends input data (images) to the first board, which processes them, running the inference of its sub-model (on the PL) then in turn passes the resulting data to the next board. Once all sub-models processed the data, the output of the last board (model classification) is returned to the client.

## III. Experimental Results

### A. Experimental Testbed

Our system is comprised of 6 Zedboards (using a Zynq XC7Z020) connected to a 100 Mbps Ethernet switch. The Zynq XC7Z020 is a heterogeneous system with a dual-core processor (Cortex-A9) coupled with an FPGA (XC7Z020-CLG484-1). Each board is set up with an identical version of Petalinux v2024.1. Each board functions as an independent server capable of being queried for the inference of a sub-model they are hosting on their PL. All boards are connected to a switch. A laptop acts as client and sends data to the first board and received the model output from the last (see Fig. 1). Time measurement has been taken client-side to account for all network transit.

### B. MobileNetV1

MobileNetV1 [6] is a CNN with 3.22 million parameters. It is quantized, 4 bits for weights and activation. FINN github lists this model as having a minimum hardware requirement of the Alveo U250. Using our splitting script, we were able to divide the DNN into 6 bitstreams that each can fit in a single XC7Z020, achieving a throughput of 44.32 frames per second. Table I shows our current results.

TABLE I
Comparison to existing work on MobileNetV1

| Work | FPGA | Number of boards | FPS | Est. total power (W) |
|---|---|---|---|---|
| Ours | Zedboard | 6 | 44.32 | 16.17 |
| [7] | UltraScale+ | 1 | 17.85 | n/a |
| [5] | Alveo U280 | 1 | 3741 | n/a |
| [5] | Alveo U280 | 2 | 5923 | 159.8 |

## IV. Conclusion

In this paper, we have presented a methodology to partition neural networks across several commodity SoC-FPGA systems to support larger models on devices with a very limited number of resources. This solution supports an arbitrary number of bitstreams and boards. Our first results are promising, supporting MobileNetV1 with a throughput of 44.32 frames per second.

Future work includes optimization of communications, partitioning larger neural networks across more FPGA chips, exploring the various trade-offs to partition such networks, *e.g.*, maximal resource usage per board, or type of resources used, *etc.*, as well as having different FPGAs contribute to the same larger neural network over a heterogeneous system.

## References


[1] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, (New York, NY, USA), p. 65–74, Association for Computing Machinery, 2017.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," Mar. 2016. arXiv:1602.02830.

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 525–542, Springer International Publishing, 2016.

[4] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[5] T. Alonso, L. Petrica, M. Ruiz, J. Petri-Koenig, Y. Umuroglu, I. Stamelos, E. Koromilas, M. Blott, and K. Vissers, "Elastic-df: Scaling performance of dnn inference in fpga clouds through automatic partitioning," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, Dec. 2021.

[6] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[7] A. Sharma, V. Singh, and A. Rani, "Implementation of CNN on Zynq based FPGA for Real-time Object Detection," *IEEE Internet of Things Journal*, pp. 1–7, July 2019.

# Compact Modelling of Reconfigurable Field Effect Transistors Towards Smart Sensing Applications

Amine Arsalane[1], Bruno Neckel Wesling[1,2], Yifan Wang[1], José Maria Gonzalez-Medina[3], Lee-Chi Hung[3], Oskar Baumgartner[3], Yuxuan He[2], Jens Trommer[2], Cristell Maneux[1], Marina Deng[1], François Marc[1] and Chhandak Mukherjee[1]

[1] IMS Laboratory, University of Bordeaux, CNRS UMR 5218, Bordeaux INP, Talence, France
[2] Namlab gGmbH, Nothnitzer Strasse 64, 01187 Dresden, Germany
[3]Global TCAD Solutions GmbH, Vienna, Austria
*Corresponding author: amine.arsalane@ims-bordeaux.fr*

***Abstract*—In this paper, we present a physics-based compact model for double-gate Reconfigurable Field-Effect Transistors (RFET). By solving the current continuity equation and corresponding charge calculations, we derive the expression of the surface potential, effective Schottky barrier and the total drain current. The proposed model demonstrates good agreement with experimental data and aims to help analog IC designers to develop RFET-based innovative circuits and smart sensing.**

***Index Terms*—Reconfigurable field effect transistor, compact modelling, Schottky barrier, drift-diffusion and drain current.**

## I. Introduction

Reconfigurable Field Effect Transistors (RFETs) are a potential key-enabling technology that can be directly co-integrated into the front-end-of-line (FEOL) of a standard CMOS process offering a number of interesting applications including analog smart sensor design. Specifically, we have chosen a 22nm FDSOI Technology for the co-integration of our RFETs. RFETs can be considered as a special variety of Schottky barrier (SB) transistors that exhibit the ability for both types of carriers conduction within a single device through electrostatic doping, thus economizing on the chip area consumed. These transistors are capable of switching between p-type and n-type characteristics at runtime, which is enabled by selecting the type of carriers tunnelling into the undoped silicon channel through the Schottky barrier based on the applied bias polarity of the polarity-control gate. In this paper, we propose a fully physics-based compact model for the carrier transport in double-gate RFET [1]. First the architecture of the RFET device under test in Section II; Section III then introduces our compact model derived from analytical equations, and finally, Section IV discusses the results obtained including model validation.

## II. Device Structure

Figure 1 schematically illustrates the structure of the double-gate RFET under test [2]. The device under test in this study was fabricated using modified I/O n-FETs from the 22 nm FDSOI technology reported in [3]. The majority carrier polarity in the device is determined by applying a voltage to the drain side gate (DG), or the polarity-control gate, while current conduction is controlled through the source side gate (SG) similar to that of a SB FET. The basic unipolar control in our device is driven by a selective injection of electrons and holes into the undoped silicon channel region as illustrated in the schematic band diagrams of Figure 2 in [1].

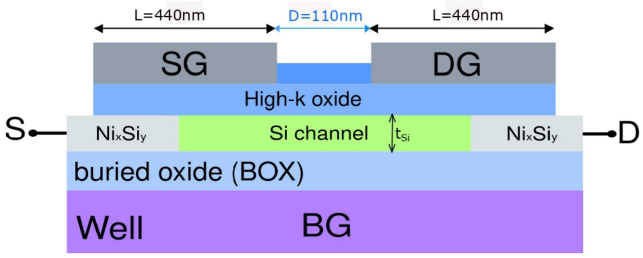



Fig. 1. Schematic of a Reconfigurable FET (RFET).

## III. Device Modelling

In this section, we introduce the organisation of our compact model. Compared to a previous compact modelling approach [4], our model derives the net drain current based on surface potential and charge calculations and the consideration of an effective Schottky barrier height. The total device current is obtained by solving the current continuity equations, which solves for the resultant current between the current through the Schottky-barrier contacts and the drift-diffusion current of both carriers in the channel. As a first step, it is essential to first solve the surface potential. To achieve this, we start with the one-dimensional Poisson equation applied to an intrinsic silicon nanowire channel [5]:

$$\frac{d^2\phi}{dr^2} + \frac{1}{r}\frac{d\phi}{dr} = \frac{qn_i}{\varepsilon_s}\left(e^{\frac{\phi - V_n}{u_T}} - e^{\frac{V_p - \phi}{u_T}}\right) \quad (1)$$

Although finding an analytical and explicit solution to equation (1) is difficult, a piecewise approach is first adopted by considering either electrons or holes, depending on the bias $V_{\text{gf}} = V_G\text{-}V_{\text{FB}}$ [5]. Moreover, by applying Gauss's law, the carrier densities respectively for the electrons and holes $Q_n$ and $Q_p$ can be obtained [2]. These expressions allow for an accurate evaluation of the surface potential and the majority carrier densities, which are crucial for determining the total

current. At the Schottky contacts, electrons (or holes) can tunnel through the Schottky barrier from the source or the drain, depending on the applied bias conditions. Two main phenomena govern carrier transport at the Schottky barrier: thermionic emission and quantum tunneling (field emission). Thermionic emission occurs when charge carriers acquire enough energy to overcome the barrier, whereas tunneling allows carriers to pass through it without exceeding the energy threshold. To simplify calculations, the tunneling probability through the Schottky barrier (SB) is set to unity if the barrier at some energy is thinner than the tunneling distance $d_t$ and zero otherwise ref [5]. To accurately model carrier transport, we define the effective Schottky barrier expression by considering both tunneling and thermionic emission effects at the drain and source sides.
The effective Schottky barrier height (SBH) for electrons at the source and drain side is given as:

$$\Phi^{S}_{\text{eff,SBn}} = \Phi_{\text{SBn}} - (1 - e^{-d_t/\lambda})(\phi_{\text{SS}} - V_{\text{bi}} - V_S) \tag{2}$$

$$\Phi^{D}_{\text{eff,SBn}} = \Phi_{\text{SBn}} - (1 - e^{-d_t/\lambda})(\phi_{\text{SD}} - V_{\text{bi}} - V_D) \tag{3}$$

Correspondingly the effective SBH for holes at the drain and source side is:

$$\Phi^{D}_{\text{eff,SBp}} = \Phi_{\text{SBp}} + (1 - e^{-d_t/\lambda})(\phi_{\text{SD}} - V_{\text{bi}} - V_D) \tag{4}$$

$$\Phi^{S}_{\text{eff,SBp}} = \Phi_{\text{SBp}} + (1 - e^{-d_t/\lambda})(\phi_{\text{SS}} - V_{\text{bi}} - V_S) \tag{5}$$

Hence, the current flowing through the barrier is expressed as:

$$I_{\text{T,n(p)}} = \pi R^2 A^*_{\text{n(p)}} T^2_{\text{dev}} \cdot \left( \exp\left( -\frac{\phi^{S}_{\text{eff,SBn(p)}}}{n_{N(P)} u_T} \right) + \exp\left( -\frac{\phi^{D}_{\text{eff,SBn(p)}}}{n_{N(P)} u_T} \right) \right) \tag{6}$$

Once the carriers have crossed the Schottky barriers, their transport within the channel is governed by the drift-diffusion model. The drift-diffusion current in the channel is obtained as [6]:

$$I_{DD,n(p)} = \mu_N(P) \cdot G \cdot \left[ 2u_T(Q_{sn(p)} - Q_{dn(p)}) + 0.5 \times \frac{(Q^2_{sn(p)} - Q^2_{dn(p)})}{C_{\text{ox}}} + Q_0 u_T \ln\left( \frac{Q_{dn(p)} + Q_0}{Q_{sn(p)} + Q_0} \right) \right] \tag{7}$$

Where $G = \frac{2\pi \cdot t_{Si}}{2 \cdot (D+L)}$. By equating drift-diffusion current and the current through the Schottky barrier, we obtain the expression for the final drain current.

## IV. Results and Discussion

Figure 3 illustrates the device transfer characteristics, comparing experimental data [2] with the compact model simulations on a semi-logarithmic scale. The figure presents the n-mode, where the drain-gate voltage (DG) acts as the polarity gate, is set to $1.8V$, while the drain voltage $V_{\text{DS}}$ varies from 0.2V to 1.8V. The same figure also depicts the p-mode, with $V_{\text{DG}} = -1.8V$ and $V_{\text{DS}}$ ranging between $-0.2V$ and $-1.8V$. The model exhibits partial agreement with experimental data but does not achieve a perfect fit. Some deviations are observed in both modes, which can be attributed to asymmetric carrier conduction and unaccounted effects such as trapping as well as limitations in the model's accuracy. Further refinements are required to enhance its predictive capability and achieve better alignment with experimental results.

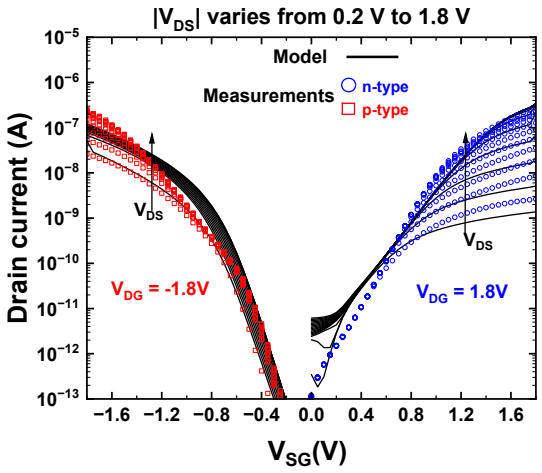



Fig. 2. Transfer characteristics ($I_D$–$V_G$) at different $V_{DS}$ for $n-type$ and $p-type$.

## V. Conclusion

This paper presents a physics-based compact model for RFETs with double gate architectures. By solving the total thermionic field emission current at the Schottky contacts and the drift-diffusion current in the channel, the total drain current is obtained. The model shows fair agreement with experimental data for different bias configurations. Further model improvement will be followed including additional effects, such as trapping.


## Acknowledgment

Funded by the European Union (Horizon Europe, SENSOTERIC, GA no. 101135316). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.


## References


[1] A. Heinzig, S. Slesazeck, F. Kreupl, T. Mikolajick, and W. M. Weber, "Reconfigurable silicon nanowire transistors," Nano Lett., vol. 12, no. 1, pp. 119–124, 2012.

[2] N. Bhattacharjee et al., "Multiple-Independent-Gate Reconfigurable FETs Processed on Industrial 300 mm FDSOI," in IEEE Electron Device Letters, doi: 10.1109/LED.2025.3549531

[3] V. Sessi, M. Simon, S. Slesazeck, M. Drescher, H. Mulaosmanovic, K. Li, R. Binder, S. Waidmann, A. Zeun, A.-S. Pawlik, et al., "Backbias reconfigurable field effect transistor: a flexible add-on functionality for 22 nm fdsoi," in 2021 Silicon Nanoelectronics Workshop (SNW), pp. 1–2, IEEE, 2021.

[4] C. Roemer et al., "Physics-Based DC Compact modelling of Schottky Barrier and Reconfigurable Field-Effect Transistors," in IEEE Journal of the Electron Devices Society, vol. 10, pp. 416-423, 2022, doi: 10.1109/JEDS.2021.3136981

[5] J. Zhang, P.-E. Gaillardon, and G. De Micheli, "A surface potential and current model for polarity-controllable silicon nanowire FETs," in Proc. ESSDERC, Sep. 2015, pp. 48–51.

[6] B. Iniguez, D. Jimenez, J. Roig et al., "Explicit continuous model for long-channel undoped surrounding gate MOSFETs," IEEE Trans. Electron Devices, vol.52, no.8, pp. 1868-1873, 2005.

# Poster: A microarchitectural signals analysis platform to craft Hardware Security Counters


1st Lucas Georget
*EDF R&D / LAAS-CNRS*
Palaiseau / Toulouse, France
lucas.georget@{edf/laas}.fr

2nd Vincent Migliore
*LAAS-CNRS*
Toulouse, France
vincent.migliore@laas.fr

3rd Vincent Nicomette
*LAAS-CNRS*
Toulouse, France
vincent.nicomette@laas.fr

4th Frédéric Silvi
*EDF R&D*
Palaiseau, France
frederic.silvi@edf.fr

5th Arthur Villard
*EDF R&D*
Palaiseau, France
arthur.villard@edf.fr



***Abstract*—Detecting malicious software or hardware behavior during the operation of a computer system requires observables from one or more abstraction layers of the system. However, this abstraction tends to limit the ability to detect behavioral deviations, especially for attack classes that exploit vulnerabilities very close to the target hardware. Conversely, too low a level of abstraction tends to significantly increase the complexity of the system model, and therefore poses a number of difficulties for the extraction and selection of relevant observables for a given class of attack. In particular, processor performance counters have been used as an indirect means of observing microarchitecture behavior and detecting software attempting to exploit hardware vulnerabilities. In order to improve the various detection methods, we propose the construction of hardware metrics designed from the outset for security, by studying the correlation between signals from the microarchitecture and the various classes of attack in the literature, targeting both usual and industrial systems. By extension, this work aims to detect attacks originating from hardware Trojans, the latter having the effect of changing the behavior of a given microarchitecture.**



***Index Terms*—Hardware Security Counters, Microarchitectural signal anaysis, Runtime monitoring.**


## I. Introduction

The detection of low-level attacks, especially hardware malware or software malware targeting microarchitectural vulnerabilities, is quite complex. Depending on the System on Chip (SoC) architecture it could turn out very differently. Hardware Performance counters (HPC) for example, can be used to trace hardware behavior and divert it for security purposes. However, the further down we go, the more complex it becomes to use observables. No hardware metrics were originally designed for security purpose.

In an increasingly complex context, where software and hardware are in close interaction, and where reconfigurable hardware architectures are becoming more and more prevalent, it is essential to be able to detect attacks with appropriate mechanisms, at the right level of abstraction. Microarchitecture signals are a good candidate for this, but it is very difficult to identify the relevant signals for detecting a specific attack. This is why it is essential to study the impact of attacks on microarchitectural signals in order to build specific counters that could be used to reference the internal state of our machines and detect attacks at microarchitectural level, as well as hardware Trojans, on both traditional and industrial equipment. A platform for capturing and analyzing microarchitectural signals, enabling a variety of experiments to be carried out, is a fundamental prerequisite. This article describes such an experimental platform, which facilitates the building of specific hardware metrics for security by studying the correlation between microarchitectural signals and different classes of attack.

The II section gives a quick overview of the state of the art. Section III then describes the platform we have designed to enable the analysis of microarchitectural signals for the definition of security hardware counters. Section IV finally proposes some perspectives to this work.

## II. Related Work

### A. Hardware Performance Counters

Hardware Performance Counters (HPC) have been used many times for security purposes. Early work [1], in the context of a fleet of IoT devices executing the same application, sought to identify deviations in the behavior of one or more devices compared to the others. An hybrid intrusion detection system [2] was created, by means of a local analysis on the devices themselves, along with a global analysis through machine learning algorithms on a remote server, in order to identify outliers in the fleet of devices. Some other research works based on learning algorithms have used HPCs to detect timing attacks on processor caches, as reported in Maria Mushtaq's thesis [3]. Against radio attacks, research works proposed the development of a monitoring and tracing system for lightweight systems [4].

### B. Hardware Signal Probing

To reach a finer level of granularity and detect even more subtle attacks, it is necessary to analyze various hardware signals and try to identify which signals are relevant to detect some specific class of attacks. This will simplify the number of measures required, as they will be directly focused on security and require less correlation. Some specific platforms, mainly based on FPGA, are necessary to carry out such experiments. But, to the best of our knowledge, few solutions are currently available for that purpose. At present, only debugging solutions such as Xilinx ChipScope and Intel SignalTap are available on the market. Mao et al [5] have been working on the instrumentation of a RocketChip in Scala to provide such a solution for software attacks that can be detected

at the microarchitectural level. Directly probing the signals associated with attack classes/families for security purposes enables them, with only very simple heuristics, to detect these attacks quickly and thus build counters with the different thresholds for each metric. But this work comes with severe constraints on the amount of data that can be retrieved.

The purpose of our research work is thus to design and implement a hardware platform, generic enough so that we can observe and exports various microarchitectural signals from the board (processor and peripherals) to perform a concrete and complete analysis and correlate this data.

## III. Microarchitectural signals monitoring platform

### A. Global view

The microarchitectural signal analysis platform should be capable of efficiently process large quantities of data. Our objective is to propose a flexible solution that captures a vector of internal signals from a System on Chip's microarchitecture with no impact on the running software. Concretely, to carry out such experiments, the solution requires:

- A reconfigurable target system, with an integrated logic analyzer for extracting microarchitectural signals.
- A host system that collects this data, with good storage capacity and good bandwidth with the target
- A high-performance system (perhaps the same as the host) to further process and analyze the data.

For a fast and portable SoC deployment into the programmable logic, we used the emerging framework LiteX [6], which is an easy way of building systems-on-a-chip, on FPGA boards. After porting the board to the project, we were able to run a small Linux system on it.

For the on-board logic analyzer, LiteScope [7] has been integrated into the SoC to provide a view of the microarchitectural signals. It is a small footprint and configurable tool able to capture signals in real time, with limited resources and without any perturbation of the system. It can be customized for our needs.

### B. Use-cases / Construction of Hardware Security Counters

The first case studies we want to experiment with this type of platform mainly concern two categories of attack:

- Software attacks, such as Cache Side-Channel and Return-Oriented Programming attacks.
- Hardware attacks at processor and peripheral levels.

The metrics collected during these various experiments will be stored and analyzed (with processing via Machine Learning in particular) in order to exhibit common detection criteria that will be used to design relevant hardware security counters. The counters could be based on thresholds of specific signals, and probably on the correlation of values of several signals according to different classes of attack. According to the scenarios, on light architectures for the industry or complex ones for computer systems, it will be possible to have different characterization proper to each hardware.

## IV. Ongoing and future work

This platform can be used for instance to improve previous work ([5]) focusing on the processor core for software attacks, by extracting relevant information relative to the instructions and memory accesses. At the moment, we only extract signals from the main CPU. As such, we can only detect malware whose behavior has an impact on these signals. We currently extend the detection logic to the signal relative to the peripherals, as it seems more suited to detect malware inserted inside the peripherals themselves or Trojan inserted at the CPU level that need to communicate with the peripherals to execute their malicious payload.

## References


[1] M. Bourdon, P.-F. Gimenez, E. Alata, *et al.*, "Hardware-performance-counters-based anomaly detection in massively deployed smart industrial devices," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, 2020, pp. 1–8. DOI: 10.1109/NCA51143.2020.9306726.

[2] N. F. Polychronou, P.-H. Thevenon, M. Puys, and V. Beroulle, "Madman: Detection of software attacks targeting hardware vulnerabilities," in *2021 24th Euromicro Conference on Digital System Design (DSD)*, 2021, pp. 355–362. DOI: 10.1109/DSD53832.2021.00060.

[3] M. Mushtaq, "Software-based Detection and Mitigation of Microarchitectural Attacks on Intel's x86 Architecture," Theses, Université de Bretagne Sud, Sep. 2019. [Online]. Available: https://theses.hal.science/tel-02988980.

[4] M. El-Bouazzati, "A Lightweight Host-based Intrusion Detection System using a Hardware-Assisted Monitor to detect Wireless Attacks Targeting Constrained IoT Devices," Theses, Université de Bretagne Sud, Dec. 2023. [Online]. Available: https://cnrs.hal.science/tel-04612764.

[5] Y. Mao, V. Migliore, and V. Nicomette, "Matana: A reconfigurable framework for runtime attack detection based on the analysis of microarchitectural signals," *Applied Sciences*, vol. 12, no. 3, 2022, ISSN: 2076-3417. DOI: 10.3390/app12031452. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1452.

[6] F. Kermarrec, S. Bourdeauducq, J.-C. L. Lann, and H. Badier, *Litex: An open-source soc builder and library based on migen python dsl*, 2020. arXiv: 2005.02506 [cs.AR]. [Online]. Available: https://arxiv.org/abs/2005.02506.

[7] EnjoyDigital, *Litescope - a small footprint and configurable embedded fpga logic analyzer*, 2015. [Online]. Available: https://github.com/enjoy-digital/litescope.

# A Comprehensive Framework for Automated and Scalable Testing of Analogue and Mixed-Signal Circuits in On-Chip Systems

Jules KOUAMO
*Univ. Grenoble Alpes, CNRS, Grenoble INP TIMA, 38000 Grenoble, France*
0009-0008-1075-5577

Emmanuel SIMEU
*Univ. Grenoble Alpes, CNRS, Grenoble INP TIMA, 38000 Grenoble, France*
0000-0001-7649-3225

Michele PORTOLAN
*IEEE Member*
*38000 Grenoble, France*
0000-0002-8284-3823

***Abstract*—Testing analogue and mixed-signal circuits presents significant challenges owing to their inherent signal variability, non-linear behavior, and the lack of universal fault models. This paper proposes an integrated framework that combines advanced software-based testing methodologies, machine learning–driven fault detection, and enhanced Built-In Self-Test (BIST) architectures to enable robust, automated on-chip testing. By employing Monte Carlo simulations, state-of-the-art signal transformation techniques, and innovative input stimulus generation (e.g., Amplitude-Modulated Pseudo-Random Multi-Level Sequences or APRMLS), the proposed system ensures high fault detection accuracy and improved test scalability while reducing production costs. Experimental case studies validate the approach, demonstrating its efficacy and resource efficiency across various circuit types.**

***Index Terms*—Analogue Circuit Testing, Mixed-Signal Systems, Automated Test Methodologies, Machine Learning, Built-In Self-Test, On-Chip Testing, APRMLS.**

## I. Introduction and Background

The rapid evolution of System-on-Chip (SoC) technologies, driven by advances in Very-Large-Scale Integration (VLSI), has integrated complex analogue and mixed-signal components into modern electronic systems. Unlike digital circuits, which benefit from structural test methodologies, analogue and mixed-signal circuits involve continuous signals and non-linear behaviors, complicating fault modeling and detection. The rising demand in high-reliability sectors such as automotive, aerospace, and industrial control underscores the need for automated, scalable, and cost-effective testing solutions.

Historically, testing analogue systems relied on manual calibration and heuristic methods, leading to high test times and variable outcomes. Recent advancements in embedded frameworks and machine learning (ML) have enabled high-precision automated fault detection. Moreover, numerous standards like IEEE 1149.1, IEEE 1500, and IEEE 1687 have improved access to circuit internals [1], but their adaptation to analogue domains remains challenging due to signal variability. Dedicated approaches have explored tolerance-based methods and ML classifiers leveraging signal transformations and Monte Carlo simulations.

Recent studies highlight the use of ML algorithms for detecting parametric faults through simulations [2], [3]. Concurrently, BIST architectures with advanced stimulus generation and output response analyzers have been developed for both linear and non-linear circuits [4], [5]. In this paper, we synthesize advancements in software methodologies, ML-based fault detection, and enhanced BIST architectures to propose a holistic framework for automated on-chip testing of analogue and mixed-signal circuits.

## II. Proposed Integrated Framework

Our proposed framework integrates three major components that together form a comprehensive solution to automate testing in on-chip environments:

### A. Advanced Software-Based Testing Methodologies

To overcome the challenges of mixed-signal testing, our approach begins by segregating the test flow into three distinct but interconnected phases:

- **Digital Testing Phase:** In this phase, conventional digital test techniques (e.g., IJTAG) are employed to ensure the controllability and observability of circuit elements.
- **Analogue Testing Phase:** Here, the emphasis is on performance evaluation through functional testing, involving the measurement of output characteristics under various input stimuli.
- **Integrated Testing Phase:** This phase bridges the digital and analogue domains by introducing test protocols, supported by PDL from IJTAG, that account for the interaction between the two, thereby facilitating a unified test strategy.

### B. Machine Learning-Driven Parametric Fault Detection

The second component of our framework leverages machine learning to address the challenges of parametric fault detection in analogue circuits. We generate extensive datasets by performing Monte Carlo simulations that introduce variations in component values. Signal transformations, including time-frequency domain analyses (e.g., via FFT and STFT), are applied to extract salient features from the resultant signals. These features are then used to train classification models that can accurately distinguish between fault-free and faulty circuit

behaviors. A novel evaluation metric, denoted as $M_{\text{test}}$, is introduced to jointly consider prediction accuracy, computational latency, and deployment complexity—particularly important for embedded applications.

### C. Enhanced Built-In Self-Test (BIST) with APRMLS

The third component enhances traditional BIST architectures by integrating an on-chip APRMLS generator (see Figure 1). This generator is capable of producing test stimuli with multiple amplitude levels, significantly improving the excitation of both linear and non-linear system responses. Coupled with an ML-based Output Response Analyzer (ORA), the improved BIST architecture reduces external dependency on high-cost test equipment, facilitates low-power operation, and supports rapid, scalable testing. This approach not only enhances fault detection accuracy but also ensures seamless integration into compact SoC designs.

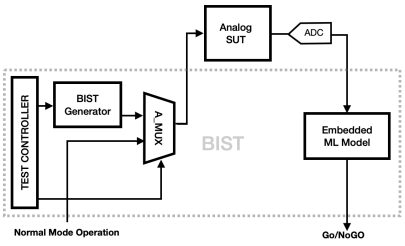



Fig. 1: Top-Level Architecture of the On-Chip BIST System

## III. Methodology

### A. Data Generation and Signal Acquisition

Data generation is achieved via Monte Carlo simulation frameworks in which circuit behavior is evaluated under a wide range of process variations. Input stimuli, such as APRMLS and conventional PRBS, are applied to the Circuit Under Test (CUT), and the corresponding output signals are sampled at rates satisfying the Nyquist criterion. The resulting dataset is labeled based on a comparison of observed performance metrics with manufacturer-specified tolerances, distinguishing between fault-free and defective circuit conditions.

### B. Machine Learning Model Development

The labeled datasets are used to train various classification models aimed at fault detection. We explore a suite of models including Random Forests, Support Vector Machines, Convolutional Neural Networks, and Recurrent Neural Networks. Key emphasis is placed on optimizing the models for deployment constraints typical of embedded systems. In this regard, the deployment-oriented metric $M_{\text{test}}$ is defined as follows:

$$M_{\text{test}} = w_1 P + w_2 \left(1 - \frac{T_{\text{pred}}}{T_{\max}}\right) + w_3 \left(1 - \frac{C_{\text{deploy}}}{C_{\max}}\right) \quad (1)$$

where $P$ is the predictive accuracy, $T_{\text{pred}}$ the prediction latency, $C_{\text{deploy}}$ the deployment complexity, and $w_1$, $w_2$, $w_3$ are weighting coefficients.

### C. On-Chip Integration and Autonomous Testing

The final stage of our methodology involves integrating the optimized ML models and APRMLS generators within a CMOS-based on-chip architecture. This integration is carefully engineered to minimize external interference, reduce signal degradation, and ensure that test operations can be performed autonomously. The system supports real-time decision making, allowing it to rapidly conclude whether a circuit meets the required operational criteria based solely on externally measurable parameters.

## IV. Experimental Results and Discussion

Our framework has been validated on benchmark circuits including RLC band-pass filters and second-order Sallen-Key filters. Detailed simulation studies demonstrate that the integrated approach achieves fault detection accuracies exceeding 99% for simpler circuits when using enhanced input stimuli and optimized ML models. Moreover, the use of APRMLS in the BIST architecture has shown significant improvements in excitation uniformity and reduction in computational overhead when compared to traditional PRBS-based methods. The experimental results also highlight the trade-offs between model complexity, deployment cost, and prediction latency—encapsulated in our $M_{\text{test}}$ metric. These findings underscore the potential of our framework for scalable and cost-effective on-chip testing, particularly in resource-constrained environments.

## V. Conclusion

This paper has presented a holistic framework that combines advanced software testing methodologies, machine learning–based fault detection, and enhanced BIST architectures to address the complexities of testing analogue and mixed-signal circuits in modern SoCs. The integrated approach not only demonstrates high accuracy in fault detection but also ensures scalability and cost efficiency by embedding critical test functionalities on-chip. Future research will focus on further refining the deployment aspects, reducing power consumption, and extending the framework to a broader range of circuit types to meet the evolving demands of next-generation electronic systems.

## References


[1] M. Portolan, M. Keim, J. Rearick, and H. Ehrenberg, "Refreshing the jtag family," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, pp. 1–7.

[2] S. Srimani and H. Rahaman, "Testing of analog circuits using statistical and machine learning techniques," in *2022 IEEE International Test Conference (ITC)*. IEEE, 2022, pp. 619–626.

[3] H.-G. Stratigopoulos and Y. Makris, "Error moderation in low-cost machine-learning-based analog/rf testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 339–351, 2008.

[4] H. Malloug, M. J. Barragan, S. Mir, E. Simeu, and H. Le-Gall, "Mostly-digital design of sinusoidal signal generators for mixed-signal bist applications using harmonic cancellation," in *2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW)*. IEEE, 2016, pp. 1–6.

[5] A. Dhayni, S. Mir, L. Rufer, and A. Bounceur, "Pseudorandom functional bist for linear and nonlinear mems," in *Proceedings of the Design Automation & Test in Europe Conference*, vol. 1. IEEE, 2006, pp. 1–6.

# Hybrid Intrusion Detection Systems for Internet of Things Gateway Against Wireless Attacks

Tianxu Li
*UMR 6285, Lab-STICC*
*Université Bretagne Sud*
Lorient, France
tianxu.li@univ-ubs.fr

Philippe Tanguy
*UMR 6285, Lab-STICC*
*Université Bretagne Sud*
Lorient, France
philippe.tanguy@univ-ubs.fr

Camille Monière
*UMR 6285, Lab-STICC*
*Université Bretagne Sud*
Lorient, France
camille.moniere@univ-ubs.fr

Guy Gogniat
*UMR 6285, Lab-STICC*
*Université Bretagne Sud*
Lorient, France
guy.gogniat@univ-ubs.fr

***Abstract*—The exponential growth of the Internet of Things (IoT) and the multiplicity of available protocols have increased the attack surface of gateways in machine-to-machine networks. Thus, early detection of attacks has become an ever-growing concern, often using time- and energy-consuming cloud computing resources, and targeting specific protocols. This paper deals with intrusion detection systems (IDS) to detect wireless attacks against IoT gateways. Indeed, given the complexity and multiprotocol nature of multi-tenant IoT gateways, traditional security solutions often fall short. We propose a novel hybrid IDS that combines software and hardware designed to provide efficient and robust intrusion detection tailored for multiprotocol IoT gateways.**

***Index Terms*—Internet of Things, IoT Security, Gateway, Wireless communication, Intrusion Detection System**

## I. Introduction

In this era of rapid digitalisation, the Internet of Things (IoT) has become an important part of our society. It is used in many key areas, such as healthcare, industry, and transportation [1]. The IoT enables diverse physical devices to collect or exchange data through networks, but also introduces new risks in wireless communications. IoT gateway, as a hub that connects these devices and transmits data to other processing systems, wireless attacks against it (e.g., jamming attack, replay attack) can affect the stability and reliability of the entire network or even lead to data leakage [2].

Intrusion Detection Systems (IDS) have been proven to be effective tools to monitor network activity, identify potential threats, and generate timely alerts for response. Hybrid IDS, in particular, with the flexibility of software and the fast responsiveness of hardware, offer efficient processing capabilities and rapid response times, making them ideal for fortifying IoT gateways.

This paper introduces our proposed hybrid IDS that is deployed on IoT gateways. It detects multiple wireless attacks against several protocols by collecting metrics from different layers of the system to enhance the security of IoT gateways.

## II. IoT gateway and IDS for IoT devices

An IoT gateway acts as a communication hub, aggregating data from various sensors and devices and transmitting it to higher-level systems. In addition, it often performs preliminary data processing tasks, such as filtering and encryption, before transmitting data for further analysis or storage [3].

The TrustGW ANR project[1] aims to enhance the security of IoT gateways by adding advanced protection features. The objective is to develop a dynamically reconfigurable and trusted heterogeneous software-hardware gateway architecture. The gateway supports both LoRa and IEEE802.15.4 wireless communication protocols, with each protocol operating independently on virtual machines managed by a hypervisor.

Within this architecture, an IDS enhances gateway security by collecting metrics from multiple layers of the system, such as the host microarchitecture and the network PHY and MAC layers. By analyzing these indicators, the IDS can detect wireless attacks in real time and issue alerts to administrators upon identifying suspicious activity. Although IDS for IoT are already discussed in the literature [4], current research is restricted to single attack types targeting single protocols [5], thus demonstrating the need for a multimodal IDS.

## III. Open research challenges

The IDS proposed in the paper is designed to detect a range of wireless attacks targeting the gateway. It adopts a hybrid implementation and is deployed directly within the gateway, enabling faster response times while conserving processor and memory resources. Moreover, the IDS must support intrusion detection across multiple wireless protocols.

The key challenges include:

- Support of multiple wireless communication protocols
- Collecting metrics from both host and network layers
- Detection of multiple wireless attack types

## IV. Proposed IDS

Figure 1 illustrates the SoC architecture of the gateway's wireless communication unit. This unit is composed of a RISC-V core (CVA6), RF modules, and our off-core hardware-based IDS component. The hardware IDS accelerates data processing and decision-making, enabling rapid response to threats while minimizing the computational load on the baseband processor. We adopted an off-core deployment strategy, as previous research has shown that this approach not only

work financed by the ANR Project TrustGW, grant ANR-21-CE39-0005

[1]https://trustgw.projects.labsticc.fr/

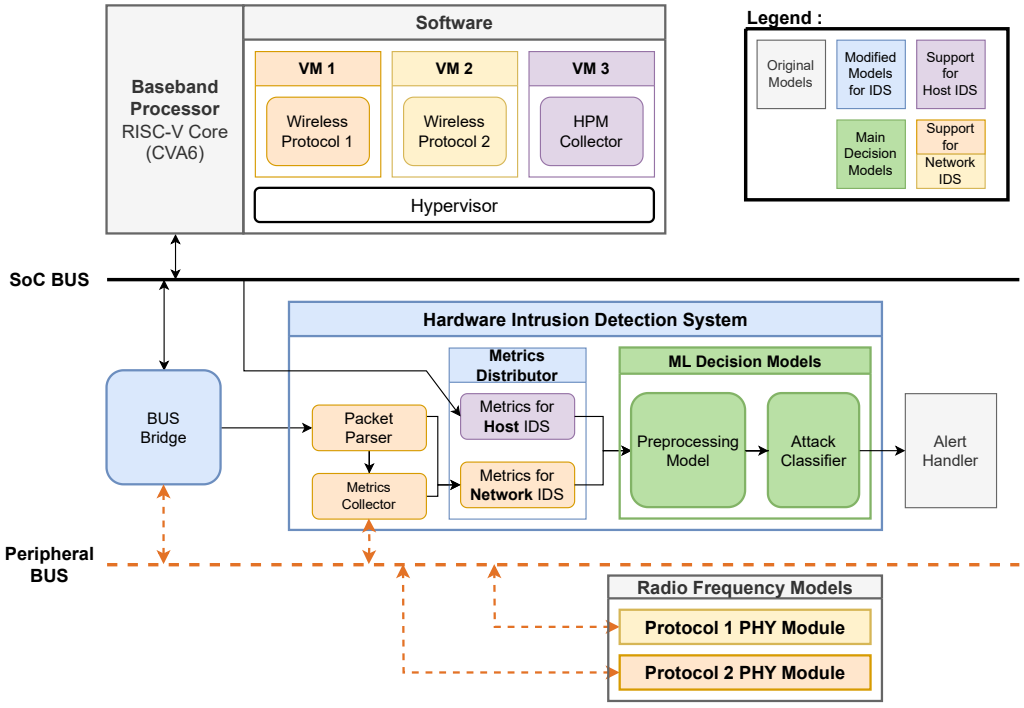



Fig. 1. Proposed hybrid IDS architecture for IoT Gateway

ensures high performances but also facilitates future reconfiguration [6]. Due to the diversity of wireless attack types, the IDS must collect metrics from multiple layers of the system.

Host-level monitoring, illustrated in purple in the figure, represents the metric collection component of the host IDS. It gathers data from Hardware Performance Monitor (HPM) to analyze the behavior of the CPU during a protocol execution.

The orange and yellow components in the figure correspond to the wireless communication subsystem, which supports Network-level monitoring. We modified the bridge between the peripheral bus and the SoC bus to intercept raw data traffic, then transmit it to the packet parser to extract PHY and MAC layer metrics. It can also request metrics directly from peripherals, without software involvement.

For metrics preprocessing, a machine learning-based approach is necessary due to the time-series nature and the high dimensionality of the collected metrics. Long Short-Term Memory (LSTM) is particularly well-suited for our scenario [7].

As for decision-making, Support Vector Machine (SVM) is suitable for classification [8]. It can be used to determine the attack's type based on the post-processed data from LSTM.

## V. Conclusion

In this paper, we propose an IDS for IoT gateways which can detect multiple types of attacks targeting different wireless communication protocols. While still in progress, the IDS has been proven to successfully identify jamming attacks against an individual node. We have also conducted preliminary tests of the HPM monitor in simulation. Currently, we are actively refining the bus bridge and the Packet parser components, and we are preparing more comprehensive datasets to train and deploy complete machine learning models.

## References


[1] S. H. Shah and I. Yaqoob, "A survey: Internet of things (IOT) technologies, applications and challenges," in *2016 IEEE Smart Energy Grid Engineering (SEGE)*, 2016. DOI: 10.1109/SEGE.2016.7589556.

[2] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, and B. Stiller, "Landscape of IoT security," *Computer Science Review*, 2022. DOI: 10.1016/j.cosrev.2022.100467.

[3] G. Beniwal and A. Singhrova, "A systematic literature review on IoT gateways," *Journal of King Saud University - Computer and Information Sciences*, 2022. DOI: 10.1016/j.jksuci.2021.11.007.

[4] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys & Tutorials*, 2019. DOI: 10.1109/COMST.2019.2896380.

[5] X.-H. Nguyen, X.-D. Nguyen, H.-H. Huynh, and K.-H. Le, "Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways," *Sensors*, 2022. DOI: 10.3390/s22020432.

[6] T. Li, M. El-Bouazzati, C. Monière, P. Tanguy, and G. Gogniat, "Comparison Between In-Core Hardware IDS, Off-Core Hardware IDS and Software IDS," in *Design and Architecture for Signal and Image Processing*, J. Lorandel and A. Kamaleldin, Eds., Cham: Springer Nature Switzerland, 2025, pp. 108–120, ISBN: 978-3-031-87897-8. DOI: 10.1007/978-3-031-87897-8_9.

[7] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM Learning With Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244–5253, Aug. 2020, ISSN: 1941-0050. DOI: 10.1109/TII.2019.2952917. (visited on 04/11/2025).

[8] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, J. Akinjobi, *et al.*, "Supervised machine learning algorithms: Classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

# Protection des SoCs par l'isolation : état de l'art et prospectives

Simon Baissat-Chavent, Lilian Bossuet, Cédric Killian
*Université Jean Monnet, UMR CNRS 5516 Laboratoire Hubert Curien*
Saint-Etienne, France

***Abstract*—L'essor des dispositifs connectés, soutenu par les systèmes sur puce (SoC), augmente le nombre de cibles disponibles à un attaquant. Les Trusted Execution Environments (TEE) isolent les opérations critiques dans un environnement sécurisé, protégé par une base de code de confiance (TCB). Ce document explore les principes d'isolation des TEE et propose une taxonomie des solutions existantes, soulignant leurs avantages et inconvénients**

***Index Terms*—Strategies d'isolation, Trusted Execution Environments, System-on-Chip**

## I. Introduction

L'électronique fait vendre et les industriels l'ont bien compris. Quel produit de la vie courante n'existe pas aujourd'hui dans une version "connectée" ? Télévisions, lampes, voitures... Certains réfrigérateurs embarquent désormais des configurations dignes de smartphones avec un OS Android complet. Or, la multiplication de ces composants intelligents implique inévitablement une augmentation significative du nombre de cibles potentielles pour un attaquant.

Ce déploiement massif de l'électronique est notamment dû à une amélioration des performances des composants, en particulier grâce à l'utilisation de systèmes sur puce hétérogènes ou SoC (System-on-Chip). Ces systèmes regroupent sur une même puce des composants de nature différente, tels que des processeurs, des mémoires, des accélérateurs matériels et des périphériques. Cependant, cette diversité de composants présente un inconvénient majeur pour la sécurité. Chaque composant introduit des vulnérabilités qui lui sont propres, et les interactions entre les composants peuvent également ouvrir de nouveaux vecteurs d'attaques. Aussi, l'effort nécessaire pour assurer la sécurité du SoC augmente avec le nombre de composants qu'il contient. En réponse à ces défis, de nombreux travaux ont proposé des solutions pour renforcer la sécurité, soit au niveau des composants individuels, soit au niveau de l'architecture globale du SoC. Une approche très répandue dans la littérature consiste à isoler les composants logiciels et matériels critiques d'un SoC (System on Chip) des autres composants du SoC qui pourraient être compromis par un adversaire. Cette approche limite ainsi la capacité d'un code malveillant à espionner un composant critique sur le SoC. Un riche écosystème de systèmes en isolation (Trusted Execution Environments ou TEE) existe, marqué par une forte dualité entre le domaine académique et le domaine commercial, ce dernier étant dominé par un petit nombre de solutions. Cette dichotomie peut, en partie, s'expliquer par le surcoût induit par le déploiement de ces solutions sur les performances générales du système.

Projet ARSENE, PEPR Cybersécurité

Dans ce document, nous présenterons le principe d'isolation sur lequel reposent les architectures TEE puis nous proposerons une taxonomie de ces architectures.

## II. Protection par isolation

Le modèle d'attaquant généralement considéré dans les travaux auxquels ce document s'intéresse consiste en un adversaire logiciel uniquement avec la capacité de corrompre l'ensemble des composants logiciels, y compris les couches privilégiées de l'OS. Un tel attaquant est donc en mesure d'accéder aux espaces mémoire d'autres programmes ou d'exécuter un code malveillant en parallèle d'un programme cible, voire de prendre le contrôle de certains drivers. Aussi, dans ce contexte, l'exécution de code critique manipulant des données sensibles, comme un algorithme de chiffrement, est très risquée, l'attaquant étant en mesure d'extraire les données sensibles telles que les clés de chiffrement.

Les architectures en isolation proposent de pallier ce problème en introduisant une ségrégation entre deux environnements d'exécution distincts. Le premier, appelé monde sécurisé, est réservé aux exécutions dépendantes de données sensibles ou de systèmes critiques du SoC (e.g. timer, stockages sécurisés, opérations cryptographiques). Le second, appelé monde normal, est dédié au reste des exécutions. Par construction, aucun code du monde normal n'est en mesure d'accéder aux exécutions du monde sécurisé. Ces architectures assument l'existence d'une base de code de confiance (Trusted Base Code ou TCB), non modifiable par l'adversaire, qui sera exécutée dans le monde sécurisé. Elle est soumise à un important processus de vérification et de validation à sa conception pour s'assurer qu'elle n'introduit aucune vulnérabilité. Aussi, l'effort de conception d'une TEE augmente significativement avec la taille de la TCB.

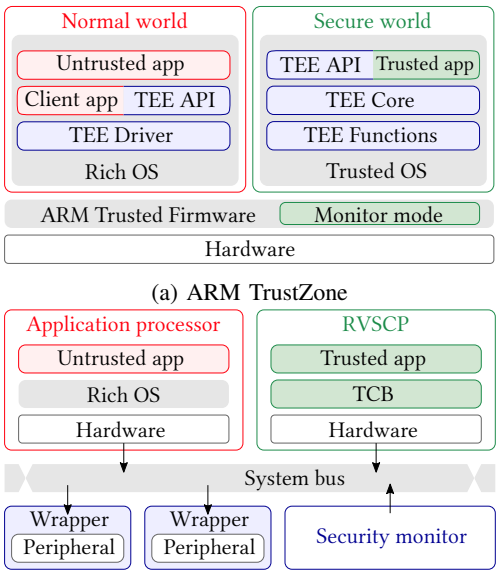
Cette ségrégation peut être implémentée de nombreuses manières, chacune avec leurs avantages et leurs faiblesses. Dans la suite, nous proposons une taxonomie des différentes solutions permettant l'isolation.

## III. Taxonomie des architectures TEE

Parmi les architectures proposées dans la littérature, on identifie deux grandes stratégies d'isolation.

La première est l'isolation logicielle avec support matériel, dans laquelle la ségrégation entre les mondes normal et sécurisé s'effectue au niveau logiciel. Les deux contextes s'exécutent l'un après l'autre sur les mêmes composants matériels (e.g. CPU) créant ainsi un partitionnement temporel du SoC. Bien que l'isolation ait lieu au niveau logiciel, ces architectures s'appuient sur des composants matériels pour assurer l'isolation. L'architecture la plus connue mettant en œuvre cette stratégie d'isolation est la solution ARM TrustZone [1], illustrée en Figure 1a. Le monde normal et le monde sécurisé s'exécutent sur les cœurs ARM, dont les changements de contexte sont contrôlés par un moniteur de sécurité dans le firmware du processeur, appelé "monitor mode". Celui-ci est chargé de nettoyer l'ensemble des ressources partagées entre les deux mondes (normal et sécurisé) comme les caches. Le monde en cours d'exécution est contrôlé par un registre dans les SCR (security configuration register) du processeur. L'architecture TrustZone est une solution commerciale proposée par ARM et fait partie des protections les plus répandues pour les SoC. Plusieurs entreprises ont créé leur propre implémentation de TrustZone, comme le QSEE de Qualcomm. SiFive propose une architecture similaire à TrustZone avec WorldGuard [7] pour leur cœur RISCV, mais introduit la possibilité de déployer plusieurs mondes normaux et sécurisés, permettant ainsi une isolation plus fine. De nombreuses solutions ont également été proposées dans le domaine académique comme Sanctuary [4] ou CURE [2].

La seconde stratégie d'isolation est l'isolation matérielle. Cette approche isole les exécutions sécurisées des exécutions normales en les exécutant sur des composants matériels dédiés. Cela conduit souvent à un modèle d'isolation plus robuste, mais au prix d'une augmentation significative du coût silicium. L'architecture HECTOR-V [6] (voir Figure b) est un très bon exemple d'isolation matérielle : l'isolation est assurée par un coprocesseur dédié appelé "RISC-V Secure Co-Processor" ou RVSCP, qui est exclusivement dédié à l'exécution du monde sécurisé. Aucune ressource micro-architecturale (cache, prédicteur de branchement...) n'est partagée entre le monde normal et le monde sécurisé, permettant ainsi de produire une isolation robuste même aux attaques sur la microarchitecture. D'autres solutions proposent une isolation matérielle exploitant la nature hétérogène des SoC, comme par exemple TrustSoC [5] . Cette architecture propose une extension d'ARM TrustZone à l'ensemble d'un SoC par le déploiement de wrapper autour des différentes IPs du SoC. Ces wrapper sont chargés de filtrer les interactions sur le SoC afin d'empêcher les accès illégaux issus de composants non sécurisés. Chaque IP est ainsi assigné à un monde et seuls les IPs du monde sécurisé sont autorisés à accéder aux données sensibles.




(a) ARM TrustZone

(b) HECTOR V

Fig. 1: Exemples de solutions d'isolation (a) logicielle avec support matériel et (b) matérielle

## IV. Conclusion et travaux futures

Il est intéressant d'observer que la quasi-totalité des solutions commerciales de TEE s'appuie sur une isolation logicielle avec support matériel. Cela est en partie dû à leur plus faible impact sur les performances du SoC comparé à une solution basée sur l'isolation matérielle. Cependant, ce choix résulte bien souvent en une plus faible isolation et plusieurs papiers ont déjà démontré la possibilité de passer outre l'isolation pour accéder à des données sensibles du monde sécurisé depuis le monde normal [3].

Le développement des Soc-FPGA ouvre de nouvelles opportunités dans le domaine des TEEs matériels en permettant le déploiement d'IPs dédiées au contrôle de l'isolation dans les cœurs du SoC. Dans nos travaux futurs, nous étudierons la possibilité d'exploiter le caractère hétérogène des SoC pour permettre le déploiement de TEE agnostiques de l'architecture des processeurs présents sur le SoC. L'objectif est de permettre le déploiement dynamique d'environnements sécurisés héritant leurs propriétés d'une IP montée sur le SoC.

## References


[1] T. Alves and D. Felton. Trustzone: Integrated hardware and software security. *ARM Inf. Q.*, 3(4):18–24, 2004.

[2] R. Bahmani, F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf. CURE: A security architecture with CUstomizable and Resilient Enclaves. In *2021 USENIX Security Symposium*, pages 1073–1090, USA, 2021. USENIX Association.

[3] E. M. Benhani, C. Marchand, A. Aubert, and L. Bossuet. On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC. In *2017 IEEE International System-on-Chip Conference (SOCC)*, pages 108–113, USA, 2017. IEEE.

[4] F. Brasser, D. Gens, P. Jauernig, A.-R. Sadeghi, and E. Stapf. SANCTUARY: ARMing TrustZone with user-space enclaves. In *2019 Symposium on Network and Distributed System Security (NDSS)*, USA, 2019. IEEE.

[5] R. Milan, L. Bossuet, L. Lagadec, C. A. Lara-Nino, and B. Colombier. TrustSoC: Light and efficient heterogeneous SoC architecture, secure-by-design. In *2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, USA, 2023. IEEE.

[6] P. Nasahl, R. Schilling, M. Werner, and S. Mangard. HECTOR-V: A heterogeneous CPU architecture for a secure RISC-V execution environment. In *2021 ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, page 187–199, USA, 2021. ACM.

[7] SiFive. Securing the RISC-V revolution : WorldGuard, 2023.

# Timing Prediction of Deep Neural Networks on Multi-Core Platforms with Memory Hierarchy Effects

Antoine CANTIN[1,2], Sebastien LE NOURS[1], Sebastien PILLEMENT[1],
Domenik HELMS[2], Kim GRÜTTNER[2], and Ralf STEMMER[2]

[1]Nantes Univ, CNRS, IETR UMR 6164, F-44000 Nantes, France, Email : antoine.cantin@etu.univ-nantes.fr
[2]German Aerospace Center (DLR), Oldenburg, Germany

***Abstract*—In the domain of edge computing, the implementation of Deep Neural Networks (DNNs) on small, resources-limited platforms is difficult. In order to optimize the inference of such systems, performance estimation is required. In that context, we propose a novel evaluation flow for mapping a DNN on embedded multi-core CPU devices. Our main contribution is to take into consideration the effect of a complex memory hierarchy on the computation time. The proposed framework is based on a SystemC simulation and calibrated with samples collected on a prototype.**

***Index Terms*—Embedded AI, non-functional properties evaluation, DNN implementation**

## I. Introduction

In recent years, the significant increase in demand for Internet-of-Things (IoT) applications and the appearance of new edge computing Machine Learning (ML) algorithms have allowed the emergence of the embedded Artificial Intelligence (AI) paradigm. One challenge is to integrate Deep Neural Networks (DNNs) on devices that have limited resources. On such platforms, the inference phase of a DNN must respect strict constraints such as energy, memory size and timing. As such, optimizations have to be conducted to make the deployment of networks on embedded devices as efficient as possible. However, the involved complexity and the massive design space for both DNN algorithms and hardware architectures make it difficult to find optimized solutions. In response to this problem, Design Space Exploration (DSE) flows were created to explore the multiple possibilities and find designs that best fulfill the requirements. In order to improve the quality of DSE, an evaluation flow is needed to predict quickly and precisely the execution time and the energy consumption of embedded platforms running DNNs.

In the literature, we identified three main methods of prediction for evaluation purposes. First, analytical modeling, where pure theory and mathematical formulas are used to model the system. Then, behavioral simulation, where the behaviors of all the components in the device are described individually and simulation tools predict how they interact with each other. And finally rapid prototyping, where measurements are performed on a prototype of the system. These methods can be compared in terms of precision, speed, scalability and required modeling effort. For example, behavioral simulation is more precise than analytical modeling because it can capture microarchitectural details, handle unpredictable events and consider complex interactions between devices. However, this accuracy comes with a lack of scalability because the simulation execution becomes heavier as the complexity of the platforms grows. A few works such as [1] also proposed hybrid methods which combine the three latter. It takes advantage of the benefits of each approach to produce fast yet accurate prediction for DSE.

In this work, we propose a new hybrid timing evaluation tool. We aim to optimize the mapping of a DNN on a multi-core CPU architecture. On Fig. 1, the DNN is partitioned into four clusters with a neuron level of granularity and each cluster is executed on the corresponding CPU. In our work, we focus on the impact of a complex memory hierarchy on the computation time to address large size DNNs. In consequence, the proposed architecture illustrated with Fig. 1 contains multiple CPUs communicating through a shared internal memory and getting data from a large-sized memory. To improve the performance of this model and limit the access to the large memory, data cache memories and DMA management are being considered in the scope of this work.

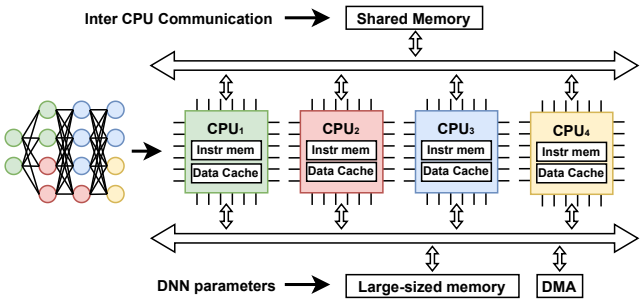



Fig. 1. Deployment of a DNN on a multi-core architecture with the considered memory hierarchy

This work is supported by France 2030 Priority research program and equipment for artificial intelligence PEPR AI, under the ref ANR-23-PEIA-0009.

## II. Related work

Several works have already developed such frameworks for architectures with multiple Processing Elements. For example, the article [1] presents a timing and power hybrid modeling flow for multi-core CPU. It is accurate (error is less than 3%) but limited in terms of memory architecture. On the other hand, a recent work [2] used simpler evaluation methods for a more complex heterogeneous architecture containing two multi-core CPUs, a GPU and a NPU. This approach aims to predict the throughput of multiple DNNs running in pipeline. Some other papers focus more on compressing the network with pruning or quantization methods. A good example is Super Slash [3]. In this article, the authors noticed that a DNN running on an accelerator can require a different number of external memory accesses depending on the chosen pruning strategy. Therefore, they created an evaluation and DSE flow for evaluating multiple pruning approaches. NetAdapt [4] propose a rapid prototyping solution. Here, direct metrics such as real latency are measured instead of proxy metrics (MACs, FLOPs, etc). The former capture the real effect of the execution of a DNN on the performance while the latter are easier to gather but less precise. The objective of this article is to compress a pre-trained DNN with an iterative hardware-aware algorithm. Also, many articles work toward optimizing the hyperparameters of DNNs and their organization. The goal is to optimize the performance without loosing too much precision. Most of these works aims for Neural Architectural Search (NAS) to find the best DNN organization possible. For instance, MAPLE [5] wants to predict hardware-aware latency for a given organization. The authors used ML to easily adapt to new hardware.

## III. Modeling and Evaluation flow

To avoid starting from scratch, our framework will be based on [1] with an update of the existing architecture to match ours. The proposed flow is described in Fig. 2. First, a pre-trained DNN model is partitioned into different clusters which are then mapped on a multi-core CPU platform. Our prototype rely on a ZCU102 FPGA implementing the architecture described in Fig. 1. Currently, the hardware design is made up of seven Microblazes and different Xilinx IPs connected through AXI buses. A Zynq processor is also used to control access to the large-sized memory. On this real platform, timing measurements are performed. The measurement infrastructure is constituted of several IPs especially designed to capture the number of clock cycles between two signals sent by the processor. The information is then sent to a host PC through an UART connection.

The evaluation flow takes three main inputs. The first one is the hyperparameters of the DNN and the grain level at which it is represented ("Partitioning" in Fig. 2). It is used to build a model of the network following the rules of the Synchronous Data Flow (SDF) model of computation. Then, the timing measurements characterize an analytical model that gives the elementary delays of communication, *Tcomm* and computation, *Tcomp*. Finally, the specifications of the platform are collected to build a model of the architecture. Everything is then integrated into a SystemC code that can predict the timing of a specific mapping on the prototyped platform. It's important to note that timing measurements are only required once. Afterwards, the model is already calibrated and new mappings can be evaluated without doing the characterization again.

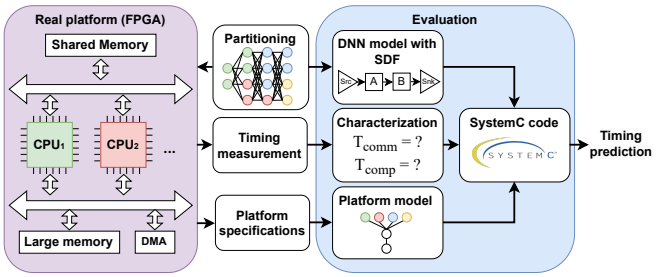



Fig. 2. Overview of the proposed modeling and evaluation flow for timing prediction of DNN mappings.

## IV. Future work

The flow, in its current state is able to predict precisely the timing of a DNN running on a multi-core CPU platform. The next step is integrating the effect of the new memory hierarchy in the evaluation software : characterizing the timings, changing the SDF model and updating the SystemC code. Also we would like to test this application with ResNet, a standard State-of-the-Art CNN for embedded devices. It would require to update the compilation workflow as the actual interface is only capable of working with MLPs and is not compatible with more complex mainstream frameworks such as PyTorch. An other future goal of this work is to be able to perform power consumption prediction. In the long term we would also incorporate more heterogeneity in the architecture by supporting hardware accelerators. Finally, in the future, it will be interesting to integrate our evaluation model into a DSE flow to find optimized mappings of the network.

## References


[1] Q. Dariol, *Early Timing and Energy Prediction and Optimization of Artificial Neural Networks on Multi-Core Platforms*. PhD Thesis, Nantes Université, Nov. 2023.

[2] E. Aghapour *et al.*, "ARM-CO-UP: ARM COoperative Utilization of Processors," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 29, pp. 86:1–86:30, Sept. 2024.

[3] H. Ahmad *et al.*, "SuperSlash: A Unified Design Space Exploration and Model Compression Methodology for Design of Deep Learning Accelerators With Reduced Off-Chip Memory Access Volume," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, pp. 4191–4204, Nov. 2020.

[4] T.-J. Yang *et al.*, "NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 285–300, arXiv, Sept. 2018. arXiv:1804.03230 [cs].

[5] S. Abbasi *et al.*, "MAPLE: Microprocessor A Priori for Latency Estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2747–2756, arXiv, May 2022. arXiv:2111.15106 [cs].

# Digital SRAM Modeling for Test

D. Ronga P. Girard A. Virazel

*LIRMM – University of Montpellier / CNRS*
Montpellier, France
dronga, girard, virazel@lirmm.fr

***Abstract*—Testing memory components is crucial to ensure the reliability of electronic devices. As the demand for memory performance and integration density continues to grow, the extreme miniaturization of the technological node provides an effective response to modern requirements. However, the miniaturization process contributes to an increased multiplicity of manufacturing defects, and an increasing complexity of certain fault mechanisms in memories. The functional test approach, which is widely used in memory testing, could show limitations in the future given the increasing memory complexity. To maintain a high level of quality in memory components, a structural testing methodology, based on the digital Cell-Aware test methodology, has been introduced. The structural test method, which moves analog memories to the digital domain from the test point of view, raises numerous challenges due to paradigm shifts. This paper presents and discusses critical points that have been studied and solved to develop a digital SRAM memory model that allows a complete implementation of the structural test methodology.**

***Keywords—Memory test, memory model, structural test.***

## I. Context and Motivations

To satisfy the performance needs of modern technologies such as Artificial Intelligence (AI), computing power requirements are constantly increasing in modern systems such as System-on-Chip (SoC). Memories, which play a central role in these systems, are required to cope with the always increasing demand of performance and integration density. The extreme miniaturization of the technological nodes in memories provides an effective response to modern requirements. However, as memory components shrink, they become more susceptible to manufacturing defects, and the complexity of certain fault mechanisms increase [1].

To guarantee the reliability of these computing systems, the reliability of memory elements must be guaranteed. Memory testing is generally based on a functional testing approach. A common approach is to use test algorithms, often March algorithms [2], due to their linear complexity, to target Functional Fault Models (FFM) [3], which represents a functional deviation from specifications. However, facing the increasing complexity in memories, and to prevent and overcome any potential limitations with functional testing, new test approaches are investigated to improve test quality.

A structural test approach has been proposed in the field of CMOS memory testing, and particularly for SRAM [4]. The structural test method is based on the digital Cell-Aware test methodology [5][6] and considers the memory structure to anticipate potential manufacturing deviations that can result in memory defects. For each of these possible defects, their detection conditions are established and listed in a structural fault dictionary, which acts as a Structural Fault Model (SFM) that is compatible with digital test tools.

The structural test approach for memories, which starts by a located defect in the memory structure to establish its detection conditions, also provides precise information on the defect location for diagnosis purposes. This test method, which relies on a digital test environment to operate, allows the use of highly efficient digital test tools, such as Automatic Test Pattern Generator (ATPG) or Fault Simulator (FS), allowing optimized and defect-specific test pattern generation or test algorithm evaluation.

However, to operate in a digital test environment, a structurally detailed, and functionally accurate digital model of memory is required. The digital model must be functionally equivalent to its analog counterpart, and its structure must allow an accurate SFM mapping to support the structural testing approach in a digital test environment. This paper presents the structural test flow, and discuss the main challenges involved in the development of an effective digital model of memory for structural testing.

## II. Digital Memory Modeling for Test Purpose

To implement the structural test methodology, a digital model of memory is required to provide an accurate structural representation of the memory that is compliant with the digital test environment. The model must allow a precise mapping of SFMs to the memory and its sub-circuits, and its digital simulation must be functionally equivalent to its analog counterpart, allowing a trustworthy evaluation of complex test sequences, such as March algorithms. The model must be able to operate in a digital test environment, requiring an equivalent and functional representation of a memorization node.

Shifting from the analog to the digital domain to represent a memory presents numerous challenges, since these domains consider different conceptual assumptions. The digital domain uses discrete time and value representation to operate, where the analog domain expects a continuous definition of time and values. The digital domain also expects some digital primitives, such as n/p-MOS to drive an information from an input to an output port, making it unidirectional primitives in the digital domain [7], while they may operate bidirectionally in the analog domain. Even if bidirectional equivalent primitives are proposed in the digital domain, these bidirectional primitives are not always compliant with digital test environments.

To guarantee an equivalent data management with the analog reference, the digital representation of the memory must deal with its bidirectional nature [8] (e.g., an information can be driven from/to a Bitcell through bidirectional SRAM bitlines). To ensure the functional equivalence of the digital model with its analog counterpart, its logic simulation must represent the analog electrical states of a memory during characteristic operations (e.g., Write or Read). Under an equivalent set of memory operations, the digital model must represent the final and stable states of the signal values that are achieved by the analog simulation through its transient electrical simulation.

In addition, specific design constructions are required to allow the digital representation of a memorization node to hold an information in a digital test environment, during the

test. These equivalent constructions are necessary to represent the memorization effect of an analog memory in a digital test environment (e.g., the complementary memorization nodes driven by an inverter loop within an analog SRAM 6T Bitcell).

The digital SRAM model must be scalable to accommodate different matrix sizes and different decoding structures. Its memorization capability must allow the consideration of the memory-array data-background in a digital test environment, allowing the generation and the evaluation of complex test sequences capable of targeting SFM, owing to ATPG and FS.

## III. Improving Memory Test Quality

The flow of the structural test methodology for memories is depicted in Fig. 1. This flow illustrates the main steps in the analog (yellow) and the digital (blue) domains, to implement the structural memory test methodology. The analog domain allows the memory description ("Analog netlist") to be electrically simulated ("Analog simulation"), providing a functional reference for the development of a digital equivalent model. It also allows the layout analysis of the memory structure ("Layout analysis"), which can guide the development of realistic defective models of the memory and its sub-circuits, that can be simulated ("Analog defective simulation") in order to develop Structural Fault Models ("SFM"). The digital domain allows the representation of a digital model of memory ("Digital netlist") to be simulated ("Digital simulation") and compared to the analog reference to guarantee its functional equivalence. The elaborated SFMs ("Fault list") can be mapped to the digital model of the memory, to automatically generate optimized test patterns ("ATPG"), targeting the SFMs as fault list. These same files can also be provided to a digital Fault Simulator ("FS") to evaluate the capability of test patterns or test algorithms that are translated into patterns ("Pattern file") in covering SFMs.

The memory model discussed in Section II has been developed using a Verilog hierarchical design approach. Its digital simulation demonstrates its functional equivalence with its analog counterpart, and its structure allows SFMs to be mapped for testing. A 4x4 SRAM case study model has been developed to evaluate the capability of March algorithms to cover a list of SFMs. To demonstrate the implementation of the structural test methodology, SFMs available in the SRAM literature [9][10] have been used as fault list. These SFMs are established from the analysis of transistor-level defects in SRAM sub-circuits (e.g., Bitcell, Write Driver, Sense Amplifier, Row and Column Decoders). The fault simulation results have led to comparative tables, demonstrating individual March algorithms capability to cover structural faults in SRAM.

To further improve the quality of memory test, new SFMs can be developed, and the methodology could be combined with Design-for-Test (DfT) approaches to further improve the memory test quality.

## Acknowledgement

This work has been funded by the French National Research Agency (ANR) under the framework of the ANR-22-CE24-0014 QUALMEM (Quality Assurance of Advanced and Emerging Memory Technologies by Using Machine Learning) project.

## References

[1] A. Bosio, L. Dilillo, P. Girard, S. Paravossoudovitch, and A. Virazel, "Advanced Test Methods for SRAMs," ISBN 978-1-4419-0938-1, *Springer*, 2009.

[2] A. J. Van de Goor, "Testing semiconductor memories: theory and practice," *John Wiley & Sons*, Inc., 1991.

[3] A. J. Van de Goor and Z. Al-Ars, "Functional memory faults: a formal notation and a taxonomy," *VLSI Test Symposium*, pp. 281–289, 2000.

[4] X. Xhafa, A. Ladhar, E. Faehn, L. Anghel, G. Di Pendina, P. Girard and A. Virazel, "On Using Cell-Aware Methodology for SRAM Bitcell Testing," *European Test* Symp., pp. 1-4, 2023.

[5] F. Hapke et al., "Cell-aware Test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1396–1409, 2014.

[6] G. Mongelli, X. Xhafa, E. Faehn, D. Robins, P. Girard and A. Virazel, "A Graph-Based Methodology for Speeding-up Cell-Aware Model Generation," *International Symposium on On-Line Testing and Robust System Design*, pp. 1-6, 2024.

[7] "IEEE Standard for SystemVerilog – Unified Hardware Design, Specification, and Verification Language," *IEEE Std 1800-2023* (Revision of IEEE Std 1800-2017), 2024.

[8] D. Ronga, X. Xhafa, E. Faehn, P. Girard, T. Vayssade, and A. Virazel, "Producing a Bidirectional ATPG Compliant Verilog-HDL Memory Model of SRAM," *Int. Conference on Design, Test & Technology of Integrated Systems*, pp. 1–6, 2024.

[9] X. Xhafa, E. Faehn, P. Girard, and A. Virazel, "SRAM Periphery Testing Using the Cell-Aware Test Methodology," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[10] X. Xhafa, E. Faehn, P. Girard and A. Virazel, "A Structural Testing Approach for SRAM Address Decoders using Cell-Aware Methodology", *Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Syst*., pp. 1-4, 2024.

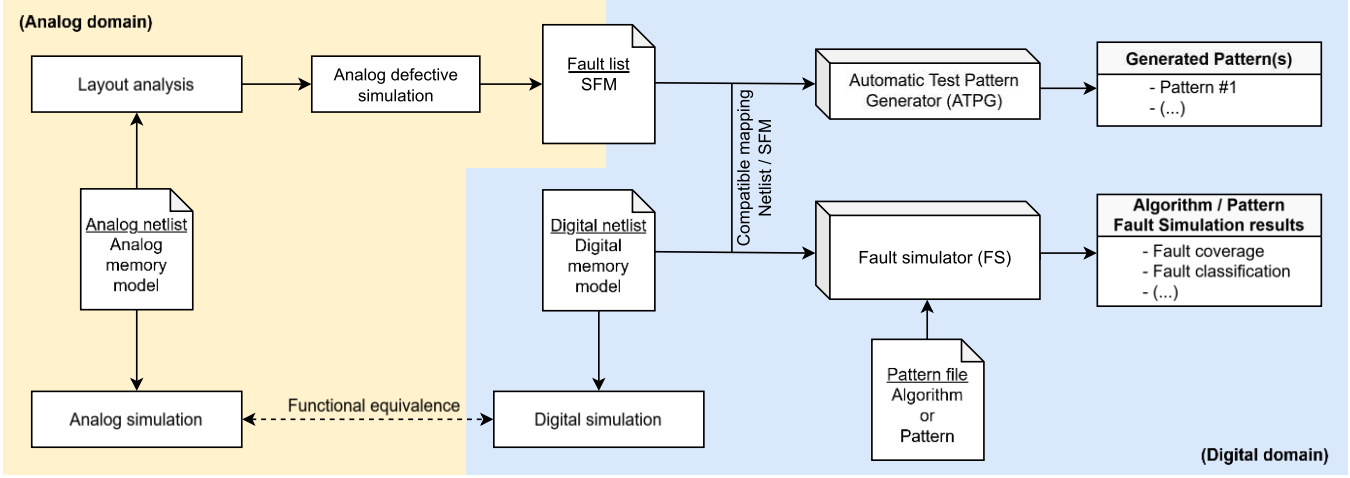



Fig. 1. Structural test flow for memories, from the analog to the digital domain

# Work in Progress: Securing a Critical Variable at Compile Time

Clara BOURGEAIS, Laure GONNORD, David HÉLY
Grenoble INP, LCIS - UGA
Valence, France
{clara.bourgeais, laure.gonnord, david.hely}@lcis.grenoble-inp.fr

***Abstract*—This article puts forward the idea of a co-design, between the hardware and the software of a computer system, combining hardware security features with code generation. We propose as a first contribution a countermeasure to a side channel attack that reduces the amount of critical information passing through vulnerable components, *i.e.* the bus and the cache. This solution consists of minimizing the spill of a critical data by integrating a pass, occurring between the instruction selection and the register allocation of a compiler. The method will be evaluated on RISC-V 32b processors.**

***Index Terms*—Physical attacks, Secure code generation, Hardware/Software co-design, RISC-V**

## I. Introduction

Hardware systems, on which software runs, are vulnerable to physical attacks. Multiple hardware vulnerabilities are regularly discovered, necessitating new countermeasures, targeting only hardware or software level.

In order to secure end-to-end a whole system, we need to co-design the hardware and the software so that: 1) hardware security properties are expressed at code generation level; 2) available hardware countermeasures are exploited; 3) software countermeasures correct remaining hardware vulnerabilities. To address these issues, we will express and ensure security properties in both the compiler back-end and the hardware during its design. We plan to develop code generation techniques through LLVM compiler back-end passes to avoid hardware-specific vulnerabilities. We also plan to formally validate co-design methodologies [1]. These objectives are part of a thesis started in October 2024, sponsored by ARSENE, a PEPR Cybersécurité project [2].

This paper proposes a first contribution for this project. We present a work-in-progress aimed at securing programs by limiting the exposure of critical information on the cache and bus during execution on a RISC-V architecture.

The paper is structured as follows: 1) Section II examines the impact of register spilling on SCA, showing that its avoidance enhances security; 2) Section III explains our use of live ranges to prevent spilling; 3) Section IV proposes a future method to validate our work;

## II. Impact of the spill on side-channel attacks

The "spill" is the compilation process of transferring a variable from a register to memory in order to release the register. This action enables the spilled register to hold a new data. During compilation, if all the registers are occupied, it becomes necessary to spill certain data (selected through heuristics) so that all the instructions can be executed properly at runtime. Spilling a variable involves moving its data through the cache or the bus, potentially exposing it to vulnerabilities, as depicted in Figure 1.

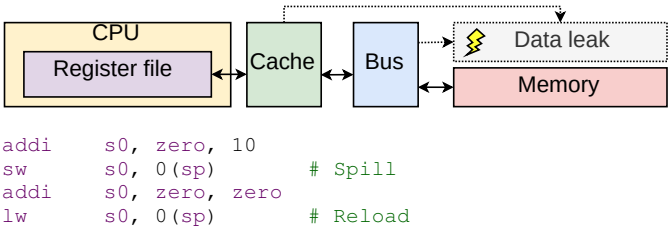



Figure 1. Illustration of a leaky register spill.

In fact, previous work has shown that both the bus and cache are vulnerable to Side-Channel Attacks (SCA) [3], [4], which are passive and non-invasive, enabling the retrieval of information (in our case, sensitive data such that -parts of- cryptographic keys), by observing auxiliary channels, like timing analysis or power consumption measurements. Such attacks can be done at relatively low-cost, with physical access to standard microcontrollers.

In this study, we thus consider that attackers can access any data passing through the bus and cache. In such cases, it is important to protect these sensitive data, or at least minimize their transactions between registers and memory. We then propose to modify the compiler spilling phase to prevent critical data for spilling.

## III. Using live-ranges to eliminate leaky spilling

### A. Observations

LLVM handles three main compilation stages: the front-end (which converts source code to intermediate representation), the middle-end (which optimizes the Intermediate Representation (IR)), and the back-end (which generates machine code). The LLVM back-end has two important passes which interests us the most [5]: 1) instruction selection, which maps IR instructions into target architecture instructions; 2) register allocation, which maps virtual registers to hardware specific physical registers, where spilling can occur if no register is available. The register

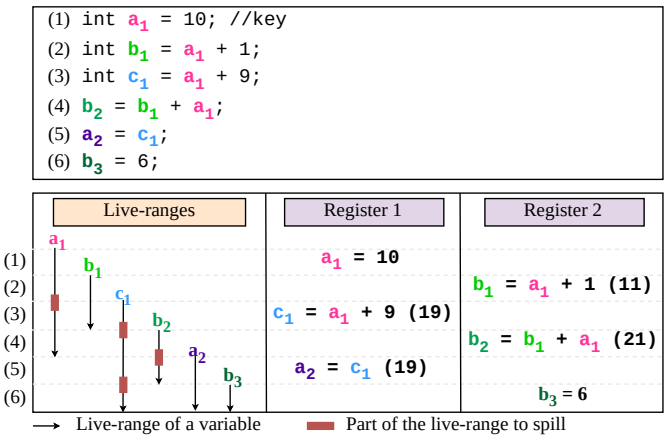



Figure 2. Illustration of live-ranges applying on a C code and impact on the register pressure.

allocator is based on early passes, particularly live-ranges analysis. A live-range represents the set of operations during which a variable remains available in a register, helping to determine the register pressure. Figure 2 illustrates how live-range analysis allows the compiler to identify cases where the number of available registers is not sufficient. For the sake of readability, the source code is still depicted in C-like syntax. It has been transformed into Single Static Assignment form [5], in which each variable is written only once, in order to ease the computation of the live ranges, depicted at the bottom left of the Figure. After line 3, $a_1$, $b_1$, $c_1$ are simultaneously*live*, requiring one to be spilled to memory. In this case $a_1$ is stored into memory (which is depicted by a brown rectangle).

However, $a_1$ being critical (as it contains a critical data), we should avoid as much as possible to prevent it for not being spilled, for all its live range.

### B. Contribution

We have consequently implemented a compiler pass in the production LLVM pipeline, executed before register allocation. Since each register in an instruction is linked to a live-range, the pass iterates over IR instructions and marks selected live ranges as non-spillable using the `markNotSpillable()` function from the live-range object.

In addition, we must selectively decide which live-ranges to mark as critical. The difficulty here is that we only have this information at the source level; potentially marked as critical by the programmer, and tracing such an information from the source code to the register allocator would be a contribution per itself. Such a tracer is for the moment also work-in-progress [6]. We thus decide to use a "known-folklore" trick.

In this work, we use inline assembly volatile instructions, added after the declaration of a critical variable, in the C source code as shown in Listing 1. The `asm volatile` keyword prevents the compiler from optimizing away [7] the `key` variable. It therefore includes a command line that cannot be optimized away. The option `"r"(key)` indicates that our `key` will we stored in a register that we can retrieve.

```
uint64_t key = 0x2b7e151628aed2a6;
asm volatile ("# key %0":: "r"(key));
```

Listing 1. Illustration of an inline volatile assembly instruction in C.

## IV. Limits and Evaluation

A non-spillable critical variable is protected, but if it depends on a spillable non-critical one, an attacker who knows the value of the latter at instruction time can infer the critical value. Therefore, all variables that influence a critical one must also be protected. Our approach should thus be combined with a technique similar to tainted flow analysis [8].

Moreover, if all variables stored in registers has to be spilled but are all critical, spilling couldn't occur anywhere else and compilation will remain impossible. We try to address this problem, by minimizing the total number of spill of these variables, *i.e.* modifying the register allocator strategy.

To check that our critical variables are not spilled and to assess the impact on the executable, we need to measure the number of spills in the assembly code and in our critical variable. A pass enabling this is under development. The aim is to be able to compare the spill rate without and with our countermeasure, and measure the impact on execution time.

We also intend to experimentally evaluate the SCA robustness of our solution on a 32-bit RISC-V microcontroller.

## V. Conclusion

In this article, we present a method for protecting a critical variable from SCA. This method consists in preventing the spill of the variable, and is part of a wider ambition to take into account hardware specificities and existing countermeasures in a hardware system.

In the future, we would like to enhance our contribution with other data-flow analyses.

We also plan to track hardware specificities more precisely by integrating hardware/software contracts at the code generation level, in order to formally prove that a system is robust against fault attacks and SCA.

## References

[1] M. Guarnieri, B. Köpf, J. Reineke, and P. Vila, "Hardware-Software Contracts for Secure Speculation," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1868–1883, May 2021.
[2] "PEPR Cyber ARSENE." https://www.pepr-cyber-arsene.fr/.
[3] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, pp. 1010–1038, June 2021.
[4] E. B. Talaki, O. Savry, M. Bouvier Des Noes, and D. Hely, "A Memory Hierarchy Protected against Side-Channel Attacks," *Cryptography*, vol. 6, p. 19, June 2022.
[5] K. D. Cooper and L. Torczon, *Engineering a Compiler*. Morgan Kaufmann, Aug. 2022.
[6] S. Michelland, "tracing-LLVM · GitLab." https://gricad-gitlab.univ-grenoble-alpes.fr/tracing-llvm/llvm, Apr. 2025.
[7] gcc.gnu.org, "Extended Asm - Using the GNU Compiler Collection (GCC)." https://gcc.gnu.org/onlinedocs/gcc-4.7.2/gcc/Extended-Asm.html.
[8] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, pp. 236–243, May 1976.

# TrustMe: A high-level cycle accurate power estimator for secure embedded processors

Mustapha Khairan Ghliss*, Yehya Nasser†, Guy Gogniat*, Salam Doumiati‡
* Lab-STICC, UMR CNRS 6285, Universite Bretagne Sud, Lorient, France
† IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
‡ L@bISEN, Usine du Futur, ISEN Yncréa Ouest, Brest, France
Email: mustapha.ghliss@univ-ubs.fr

***Abstract*—In modern digital systems, securing sensitive information goes beyond software defenses. Side-channel attacks (SCAs) exploit unintended physical emissions—such as power consumption variations—to extract secret information like encryption keys, posing a serious risk to the security of embedded systems. This vulnerability is particularly concerning in IoT networks, satellite communications, and encrypted radio systems, where data confidentiality is crucial. Traditional security measures often overlook physical-layer threats, leaving systems exposed to potential breaches. To address this gap, the TrustMe project introduces an intelligent power estimator designed to detect and mitigate cryptographic leaks at design time, enhancing security at the hardware level before attackers can exploit these weaknesses.**

***Index Terms*—Power estimation, Processor security, Artificial intelligence and Cycle accurate precision.**

## I. Introduction

As embedded systems become increasingly prevalent in applications with stringent security requirements, ensuring their resilience against security vulnerabilities is critical. Among these threats, power side-channel attacks (SCAs) pose a particularly serious concern like mentioned in [1]. As many research works pointed out, software style impacts program performance, power and energy consumption of computing devices in significant ways [2]. Furthermore, software implementations, if not handle with care, also have a significant impact regarding security. This challenge underscores the need for power-aware security evaluations early in the design process. However, existing approaches often rely on computationally intensive simulations or physical prototypes, making them impractical for early-stage analysis.

We introduce TrustMe, a high-level, cycle-accurate power estimation framework for secure embedded processors. TrustMe provides early-stage power analysis by estimating the power consumption based on instruction execution and microarchitectural behavior, balancing accuracy and efficiency. This work targets a fine level of granularity, few prior studies have addressed power estimation at the cycle-accurate level as discussed in Section II. Such a fine level of granularity allows designers to assess power leakage risks and evaluate countermeasures before a physical implementation. Our methodology involves instruction-level power characterization with cycle-accurate precision, modeling microarchitectural contributions, and integrating these elements into a simulation framework to generate per-cycle power estimates. The goal is to provide a clear, developer-friendly tool that aids in identifying vulnerable locations within software. The following sections present the state of the art and detail the methodology employed in TrustMe.

The authors would like to thank the Direction Générale de l'Armement (DGA) and the CMA CyberSkills4All initiative for their sponsorship and support. Their contributions have been instrumental in the development of this research.

## II. Related Works

Previous research in power estimation and side-channel vulnerability assessment has approached the problem from several valuable, yet ultimately limited, perspectives. At one end of the spectrum, Breier et al. [3] conducted a comprehensive RISC CPU analysis using cycle-accurate measurements at the gate level. While highly precise, this approach operates at a low level of abstraction, making it both costly and less accessible for early-stage design intervention. In contrast, Zhang et al. [4] introduced a ChipWhisperer-based methodology that demonstrated gold-standard accuracy in detecting power leaks through physical measurements. This work inspired our hardware setup for TrustMe, although such physical analyses can only identify vulnerabilities post-silicon, limiting their utility in pre-silicon design phases.

On the other end of the spectrum, machine learning-based approaches such as the FPGA/ASIC estimator in [5] and the microarchitecture-level model in [6] offer high-level abstractions and rapid analysis capabilities. However, these methods suffer from critical drawbacks; they tend to focus on average power consumption, overlooking the instantaneous leakage patterns that are crucial for side-channel analysis.

Architectural simulators such as the instruction-level RISC-V model from [7] and the ARCHER framework [8] represent significant progress in pre-silicon analysis. Yet, these models rely on fixed mathematical abstractions, which limits their scalability and adaptability. Similarly, DeepPM [9] introduces a novel transformer-based model, but operates at a high level of abstraction, estimating only the average power of a program without preserving cycle-level granularity.

TrustMe synthesizes the strengths of these diverse approaches while addressing their limitations. We preserve the cycle-accurate precision of physical measurement techniques while shifting the analysis to the critical pre-silicon phase.

Our framework avoids the opacity of pure machine learning solutions through interpretable, physically-grounded power modeling. Furthermore, it is purpose-built for security analysis which enables designers to efficiently identify and mitigate power side-channel vulnerabilities earlier in the design process thanks to a real-world leakage behavior.

## III. Methodology

The TrustMe simulator will enable developers and researchers to visualize how code-level behavior translates into power consumption, ultimately helping identify potential security vulnerabilities such as side-channel leakage. An illustrative example of how TrustMe is applied is shown in Figure 1.

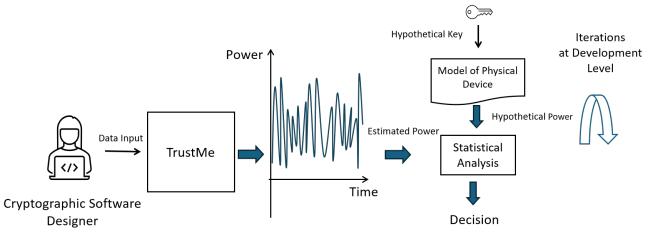



Fig. 1. TrustMe use case

TrustMe follows a structured and modular methodology to achieve precise and scalable power estimation. The core steps of our approach are as follows:

- **Dynamic Instruction Execution** – We introduce randomized execution parameters to evaluate a wide range of instruction sequences. This ensures that our analysis captures a wide spectrum of power consumption behaviors, accounting for instruction-level diversity.
- **Power Trace Collection** – We interface with a RISC-V-based hardware platform to capture real-time power traces during execution (Figure 2). These fine-grained measurements reveal subtle power fluctuations associated with specific instructions and data patterns.
- **Dataset Generation and Power Modeling** – The collected traces are processed to create a rich dataset that aids in training models capable of understanding complex correlations between instruction execution and power signatures. Leveraging this dataset, the model undergoes a dedicated modeling phase, during which it acquires pattern knowledge by learning from power consumption behavior.
- **Smart Model Optimization** – Our machine learning pipeline will include iterative refinement to improve the precision and generalization of the power model. This ensures that the system can effectively identify anomalies or potentially vulnerable instruction sequences.

The TrustMe project leverages a combination of hardware instrumentation and AI techniques, as depicted in Figure 2. This integration allows developers to not only estimate power consumption at a fine-grained level, but also to interpret results in a human-readable manner, enabling effective vulnerability localization and software hardening.

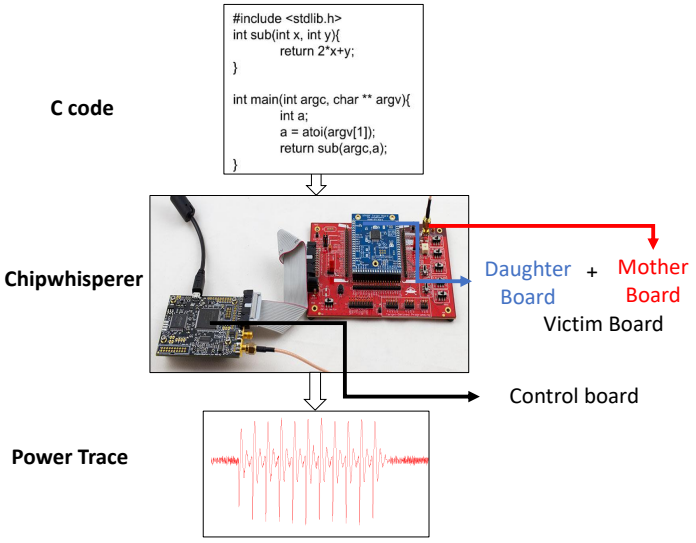



Fig. 2. Hardware setup

## IV. Conclusion

We introduced TrustMe, a power estimation framework that combines real measurements with AI-based prediction techniques to provide cycle-accurate power estimations. By enabling detailed detection of power leakage vulnerabilities, TrustMe contributes to securing embedded processors against side-channel attacks.

## References


[1] Mark Randolph and William Diehl, "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman," *Cryptography*, vol. 4, p. 15, May 2020.

[2] Wellisson G. P. da Silva, Lisane Brisolara, Ulisses B. Corrêa, and Luigi Carro, "Evaluation of the impact of code refactoring on embedded software efficiency," in *Proceedings of the 1st Workshop de Sistemas Embarcados*, pp. 145–150, 2010.

[3] J. Breier, D. Jap, C. Chen, and L. Kriege, "A Comprehensive Side-Channel Leakage Analysis of an In-Order RISC CPU Microarchitecture," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1187–1200, Aug. 2021.

[4] C. Zhang, J. Yang, and W. Zhang, "Cycle-Accurate Power Side-Channel Analysis Using the ChipWhisperer: A Case Study on Gaussian Sampling," in *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2022, pp. 1987–2004.

[5] Y. Wang, Q. Liu, J. Wang, and L. Cheng, "An Efficient Computer-Aided Design Methodology for FPGA/ASIC High-Level Power Estimation Based on Machine Learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 5, pp. 601–614, May 2022.

[6] H. Li, K. O'Brien, and A. Rahimi, "Machine Learning-Based Microarchitecture-Level Power Modeling of CPUs," in *Proc. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2020, pp. 84–95.

[7] T. Suzuki, M. Nagata, and A. Schaumont, "Instruction-Level Power Consumption Simulator for Modeling Simple Timing and Power Side Channels in a 32-bit RISC-V Micro-Processor," in *Proc. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2019, pp. 183–186.

[8] M. Yan, J. Xu, and Q. Xu, "ARCHER: Architecture-Level Simulator for Side-Channel Analysis in RISC-V Processors," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2021, pp. 1–9.

[9] K. Tang, Y. Liu, and P. Huang, "DeepPM: Transformer-based Power and Performance Prediction for Energy-Aware Software," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 3, pp. 567–580, July-Sept. 2022.

# Multi-mode Network-on-Chip for AI dataflow accelerators

Mohamed Amine Zhiri*‡, Hana Krichene*, Chiara Sandionigi†, Sébastien Pillement‡
*Université Paris-Saclay, CEA, LIST, 91120 Palaiseau, France
{mohamed-amine.zhiri,hana.krichene}@cea.fr
†Université Grenoble Alpes, CEA, LIST, 38000 Grenoble, France
chiara.sandionigi@cea.fr
‡Nantes Université, CNRS, IETR, UMR 6164, 44000 Nantes, France
{mohamed-amine.zhiri,sebastien.pillement}@univ-nantes.fr

***Abstract*—Fully Connected (FC) layers are a bottleneck for many Deep Neural Networks (DNN) algorithms due to their high bandwidth requirements, which makes their hardware acceleration particularly challenging. In this paper, we propose HT-NoC (High Throughput Network-on-Chip), a multi-mode NoC to accelerate FC layers. Compared to a baseline mesh, HT-NoC achieves a $4\times$ reduction in latency and a $2.7\times$ decrease in energy consumption in the propagation of FC layer weight parameters. When integrated into an AI dataflow accelerator, HT-NoC achieves a $3\times$ speedup in executing Feed Forward Network (FFN) blocks in Transformers, outperforming state-of-the-art (SoA) systolic array (SA) based accelerators.**

***Index Terms*—Network-on-Chip, AI accelerators.**

## I. Introduction

DNNs have recently shown enormous potential, with architectures such as convolutional neural networks and transformers leading the way in fields like computer vision and natural language processing. A common layer among those two architectures is the FC layer. FC layers are bandwidth bound, hence efficient communication is critical for their acceleration. In this work, we propose HT-NoC, a multi-mode NoC that dynamically adjusts its throughput to match the bandwidth needs of FC layers. To achieve this, HT-NoC uses a reconfiguration mechanism that involves rerouting router ports to fully utilize unused NoC resources thereby taking advantage of available unused bandwidth. HT-NoC shows promising results for some Convolutional (CONV) layers also.

## II. Related Work

Two different strategies can be used to increase bandwidth and reduce latency in NoCs and routers namely : resource duplication and resource usage. Resource duplication solutions such as widening links [1] or multiple routers [2] can increase overall performance but incur higher area and power overheads. These overheads can be overcome using resource reuse strategies [3] [4]. However, the performance gains from reuse can be limited if only certain components are optimized—for instance, improving buffers without addressing channel inefficiencies, or vice versa, may constrain the overall benefits. Therefore, for optimal results, it is necessary to reuse all available NoC resources. Our strategy in HT-NoC revolves around adding limited amount of logic to fully reuse all available router resources.

## III. Architecture and data propagation

HT-NoC features a 2D-mesh topology and utilizes the XY routing algorithm alongside wormhole flow control, with a channel width of 32 bits. Each router can operate in one of two distinct modes that are Normal Mode and HT mode. The Normal Mode serves as the default mode in which each router is directly connected to its four neighboring routers. In this mode, the router's available output bandwidth is evenly distributed across all its output ports. The HT Mode is designed for high-bandwidth traffic patterns that maintain a consistent direction. A prime example of this is the propagation of weights in FC layers. To fully utilize the available bandwidth of each router, unused output ports are reconfigured to transmit data in the same direction (west to east for our usecase). As illustrated in Fig. 1, HT blocks are switches that enable each router to switch from one operating mode to another. Black arrows demonstrate normal mode connections while blue arrows are used in HT mode. Red arrows are used in both modes.

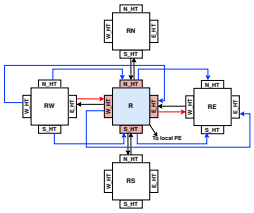


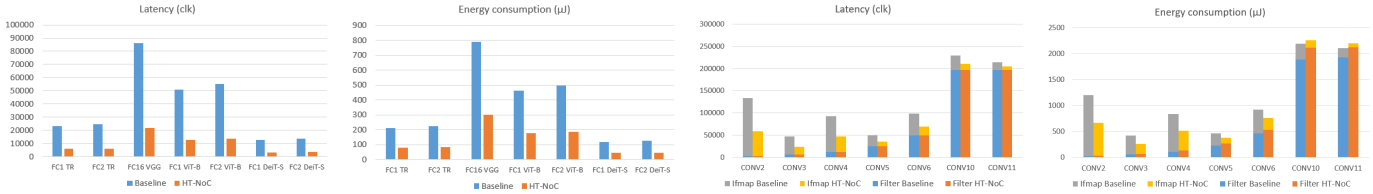

Fig. 1: Router connections

The execution of FC and CONV follows 4 different phases: filter propagation, input feature map (ifmap) propagation, computation, and output feature map (ofmap) collection. Filter propagation in FC layers and ifmap propagation in CONV layers are performed in HT mode to take profit from available bandwidth. Other phases are performed in the normal mode.

## IV. Experimental results

### A. Area

We present the FPGA synthesis results for a 12x12 HT-NoC configuration operating at 100 MHz targeting the AMD Versal



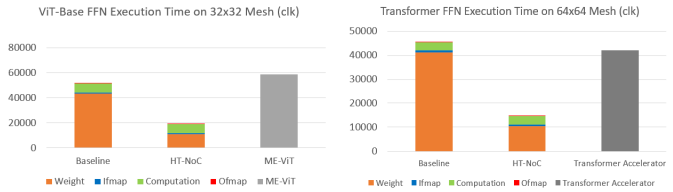

FC latency in clock cycles | FC dynamic energy in ($\mu$J) | CONV latency in clock cycles | CONV dynamic energy in ($\mu$J)

Fig. 2: Propagation of FC and CONV layers input parameters latency and dynamic energy consumption

XCVC1902 device in Tab. I. The results show that HT-NoC increased LUT usage by 17% and Flip Flops by 4% compared to the baseline NoC.

TABLE I: Hardware resource utilization

| | **Baseline router** | **HT router** | **Baseline NoC (12x12)** | **HT_NoC (12x12)** |
|---|---|---|---|---|
| **CLB LUTs** | 2064 | 2137 | 290810 | 339549 |
| **Flip Flops** | 1142 | 1146 | 158620 | 164996 |




(a) ViT-Base in 32x32 (b) Transformer in 64x64

Fig. 3: Execution time breakdown of FFN layers of Vit-Base, transformer over different configurations of the accelerator

### *B. Input data propagation*

Fig. 2 illustrates input data propagation time and energy consumption for benchmarked FC and CONV layers. Regarding latency, HT-NoC speeds up data propagation by a factor of four for all layers, compared to the baseline NoC. Energy-wise, HT-NoC achieves an average reduction of 2.7× for each FC layer. For all CONV layers, filter propagation has the same latency in both NoCs, since HT-NoC does not accelerate the filter data sending. For ifmaps, HT-NoC achieves a $2.35\times$ acceleration and an average energy saving factor of 2 compared to the baseline NoC. Unlike FC layers, the performance of HT-NoC for CONV layers depends widely on the layer. Better performance are obtained in early layers. In fact, in deep layers, filters are larger than ifmaps.

### *C. Integration into an AI accelerator*

We integrate HT-NoC into AI dataflow accelerator used in [5] to study the execution time of ViT-Base [6] and the original Transformer [7] models on the accelerator with both the baseline NoC and HT-NoC. Then we compare with the performance of Me-ViT [8] and the Transformer accelerator [9] that are both SA based architectures. Fig. 3 shows that compared to the baseline NoC, HT-NoC induced an acceleration of the weight propagation phase by a $4\times$ factor. Finally, HT-NoC delivers a speedup of $2.7\times$ and $3\times$ compared to the $32 \times 32$ baseline accelerator and Me-ViT, respectively, when executing the FFN block of the ViT-Base model. Furthermore, the FFN block of the original Transformer model was accelerated by of $3.1\times$ and $2.8\times$ over the $64 \times 64$ baseline accelerator and the Transformer accelerator.

## V. Conclusion

In this paper, we presented HT-NoC, a reconfigurable NoC that adjusts its throughput to accelerate the execution of FC layers. HT-NoC achieves a $4\times$ speedup and a $2.7\times$ reduction in energy consumption when transmitting weights of FC layers. It also achieves favorable results for some CONV layers. Lastly, we integrated HT-NoC into an AI dataflow accelerator and showed that HT-NoC can accelerate the execution time of FNN blocks of transformers by a $3\times$ factor, thus yielding favorable results compared to SoA SA-based solutions.

## References


[1] T. Fischer, M. Rogenmoser, M. Cavalcante, F.K. Gürkaynak, and L. Benini. Floonoc: A multi-tb/s wide noc for heterogeneous axi4 traffic. *IEEE Design Test*, 40(6):7–17, 2023.

[2] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[3] Y.-C. Lan, S.-H. Lo, Y.-C. Lin, Y.-H. Hu, and S.-J. Chen. Binoc: A bidirectional noc architecture with dynamic self-reconfigurable channel. In *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 266–275, 2009.

[4] H. Farrokhbakht, H. Kao, and N.E. Jerger. Ubernoc: unified buffer power-efficient router for network-on-chip. In *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '19, New York, NY, USA, 2019. Association for Computing Machinery.

[5] H. Krichene, R. Prasad, and A. Mouhagir. Ainoc: New interconnect for future deep neural network accelerators. In *Design and Architecture for Signal and Image Processing - 16th International Workshop, DASIP 2023, Toulouse, France, January 16-18, 2023, Proceedings*, volume 13879 of *Lecture Notes in Computer Science*, pages 55–69. Springer, 2023.

[6] A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[7] A. Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[8] K. Marino, P. Zhang, and V.K. Prasanna. Me- vit: A single-load memory-efficient fpga accelerator for vision transformers. In *2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 213–223, 2023.

[9] S. Lu, M. Wang, S. Liang, J. Lin, and Z. Wang. Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer. In *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, pages 84–89, 2020.

# Machine Learning Approaches for Application Mapping Optimization in Network-on-Chip Architectures

Zainab GHRAYEB[1,2], Sebastien LE NOURS[2], Christophe MOY[1], Jordane LORANDEL[1], and Christine SINOQUET[3]

[1] Univ Rennes, CNRS, IETR UMR 6164, F-35000, Rennes, France Email: zainab.ghrayeb@etudiant.univ-rennes.fr
[2] Nantes Univ, CNRS, IETR UMR 6164, F-44000 Nantes, France
[3] Nantes Univ, CNRS, LS2N UMR CNRS 6004, F-44000 Nantes, France

***Abstract*—Application mapping optimization is a critical challenge in Network-on-Chip (NoC) design, as heuristic methods often struggle to find optimal solutions. Various new mapping frameworks have been proposed, recently employing different predictive machine learning models and search strategies. However, a comparison of these approaches reveals a common limitation: they often overlook interactions with the memory hierarchy. To address this gap, we propose an adaptation of existing frameworks to support more complex NoC architectures, taking memory interactions into account for studying the effect on performance and scalability.**

***Index Terms*—application mapping, network-on-chip (NoC), predictive model, search approach.**

## I. Introduction

Currently, network-on-chip (NoC) represents the main solution for efficient communication infrastructure in multiprocessor systems. NoCs provide high-performance, scalable, efficient, and high-bandwidth communication between cores, overcoming the limitations of traditional bus-based interconnects. NoC systems differ from each other depending on topology (e.g., 2D mesh, torus, 3D mesh), routing algorithms, latency, throughput, power consumption, and scalability. Designing NoCs is particularly complex due to the large number of interdependent parameters (like buffer size, number of virtual channels, or routing method). Each design choice affects multiple performance and energy metrics, often in conflicting ways.

In addition to these architecture challenges, the application mapping adds another layer of complexity. Application mapping involves assigning application software tasks to processing elements connected through the NoC. The objective is to minimize the communication latency and energy consumption by balancing computation and communication load distribution. Poor mapping can negate the benefits of a well-designed architecture, making efficient mapping strategies critical for overall system performance.

Some existing traditional mapping algorithms rely on exhaustive search methods, but they suffer from reaching optima in a prohibitive search duration. Machine learning (ML) techniques have been introduced for their ability to favor mapping decisions and reduce exploration time. These algorithms have the ability to extract different relations between tasks of an application, guide the search for the optimal solution, and handle large-scale mapping. From this scope, in our work, we plan to adopt machine learning algorithms to favor the static mapping optimization problem for some specific NoC architectures.

This work is supported by France 2030 Priority research program and equipment for artificial intelligence PEPR IA-Projet AdaptING, under the ref ANR-23-PEIA-0009.

## II. Related Work

Over the years, various algorithms have been proposed for the static mapping problem of dataflow oriented applications. Static mapping methods include exact techniques like *mathematical programming* or *search-based mapping*. Mixed-integer linear programming (MILP) has been used with clustering to reduce complexity, though this often compromises solution quality [1]. Since most previous methods target small IP-core systems, *heuristic search* has become the preferred approach for large-scale mappings. Heuristic methods are generally classified into transformative and constructive heuristics.

*Transformative heuristics* use evolutionary algorithms to explore the solution space, but they rely on stochastic processes that may suffer from poor scalability [2]. *Constructive heuristics* build mappings step by step, either incrementally without improvement or with an iterative refinement phase. However, if the initial construction criteria are not well-designed, the final mapping may be suboptimal [2].

In recent years, ML have been increasingly applied to solve mapping optimization problems as illustrated in Fig. 1. As illustrated, ML-based predictive models typically take two inputs: an application graph, representing tasks and their communication, and the Network-on-Chip (NoC) architecture, consisting of a defined number of processing elements (PEs). Table I highlights some articles that propose ML-based predictive models. These models iteratively compute the probability of each candidate mapping until the specified number of mappings is reached. From the obtained mapping solutions, different search approaches are applied to explore the mappings further. For every solution, the communication cost is computed, and the mapping with the lowest cost is ultimately selected as the final mapping decision for the given

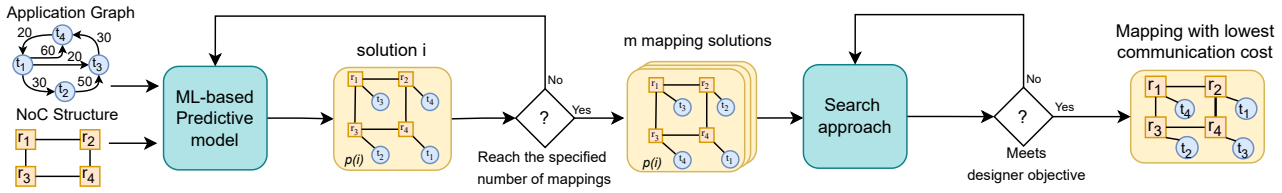



Fig. 1. General illustration of the mapping process. This flow aims to illustrate existing work for the application mapping problem. Refer to Tab. I for detailed information for each step.

application and platform. These mapping processes differ depending on the predictive model and the convergence speed of the search method.

*Chen et al* in [1] proposed the MPNN-Pointer Network (MPNN-PtrNet), a two-phase ML-based predictive model designed for application mapping. In the first phase the application graph is processed using a Message Passing Neural Network (MPNN) to extract high-level structural features. In the second phase, an attention mechanism is employed to select elements from the input sequence as outputs, effectively generating the mapping solution. This proposed predictive model was used in other papers [2], [3] but differ in the search approach performed. *Sambangi et al* in [4] propose another predictive model composed of Graph Attention Network (GAT) that uses an attention mechanism to weight the contributions of neighboring nodes followed by a Ptr network, to extract more features from the graph application that improve the mapping solutions especially if working with 3D NoC instead of 2D NoC. Optimizing communication cost for a 2D NoC, the reinforcement learning-based method proposed in [2] demonstrates a noticeable reduction in communication energy compared to the supervised learning approach in [1] and traditional heuristic search methods. Also this method shows a reduction in the CPU execution time compared to traditional methods. Furthermore, replacing the search mechanisms used in [2] with the active search strategy proposed in [3] leads to an even greater reduction in communication cost, highlighting the effectiveness of adaptive and reward-guided exploration. While these three algorithms focus on 2D NoC architectures, the method presented in [4] extends the application to a 3D NoC, achieving additional improvements in both communication cost and energy efficiency by incorporating Through-silicon Via (TSV) placement and congestion-aware mapping.

Despite their promising results, the proposed ML-based mapping methods have several limitations. Reinforcement learning approaches often involve high computational overhead during training. Early models primarily optimize the communication cost of mapping the tasks to the processing elements without accounting the communication between the processing elements and the memory hierarchy. Scalability remains a concern, especially for large application graphs, and some models require retraining to adapt to different NoC topologies. These challenges highlight the need for more efficient and generalizable solutions.

TABLE I
RECENT ML-BASED APPLICATION MAPPING APPROACHES FOR NOC.

| Paper | ML-based Predictive Model | Search Approach | Training |
|---|---|---|---|
| [1] | MPNN-Ptr | Global search followed by 2-opt local search | Supervised learning |
| [2] | MPNN-Ptr | Genetic algorithm | RL |
| [2] | MPNN-Ptr | Particle swarm optimization | RL |
| [3] | MPNN-Ptr | Active search | RL |
| [4] | GAT-Ptr | Updating reward of RL | RL |

## III. PROPOSAL

Based on the limitations of the proposed method, one promising direction involves extending the presented framework in Fig. 1, addressing the interaction with the memory hierarchy. Applications such as artificial intelligence often involve a large number of parameters that exceed the capacity of the private memory available in individual processing elements. To overcome this limitation, sharing resources between tiles and leveraging memory hierarchy can offer an effective solution. Consequently, extending the platform architecture emerges as a compelling strategy to enhance and adapt existing algorithms to meet these demands.

## IV. CONCLUSION

In this paper, we have presented the adoption of ML-based approaches for application mapping optimization in NoC architectures. Our aim is to extend the considered related works to consider more the influence of large size memory accesses in the mapping process.

## REFERENCES

[1] Chen, Qingkun, Wenjin Huang, Yuanshan Zhang, and Yihua Huang. "An IP core mapping algorithm based on neural networks." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 29, no. 1 (2020): 189-202.

[2] Chen, Q., Huang, W., Peng, Y. and Huang, Y., 2021. A reinforcement learning-based framework for solving the IP mapping problem. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(9), pp.1638-1651.

[3] Sambangi, Ramesh, Arun Sammit Pandey, Kanchan Manna, Sudipta Mahapatra, and Santanu Chattopadhyay. "Application mapping onto manycore processor architectures using active search framework." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 31, no. 6 (2023): 789-801.

[4] Ramesh, Sambangi, Kanchan Manna, Vinay Chakravarthi Gogineni, Santanu Chattopadhyay, and Sudipta Mahapatra. "Congestion-aware vertical link placement and application mapping onto three-dimensional network-on-chip architectures." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2024).

# Adaptative and optimized AI for the IoT Edge Cloud Continuum

1st Léo BERNARD
*Université Côte d'Azur, CNRS, LEAT*
*Sophia Antipolis, France*
*leo.bernard@univ-cotedazur.fr*

2nd Alain Pegatoquet
*Université Côte d'Azur, CNRS, LEAT*
*Sophia Antipolis, France*
*alain.pegatoquet@univ-cotedazur.fr*

3rd Laurent Rodriguez
*Université Côte d'Azur, CNRS, LEAT*
*Sophia Antipolis, France*
*laurent.rodriguez@univ-cotedazur.fr*

***Abstract*—The increasing deployment of low-power IoT sensors has driven the need for more efficient data processing paradigms that go beyond the traditional cloud-centric model. The IoT-Edge-Cloud Continuum (IECC) emerges as a promising architecture, unifying the strengths of IoT devices, edge nodes, and cloud servers to enable real-time, energy-efficient applications. However, the heterogeneity, dynamic nature, and multi-objective optimization challenges inherent in this continuum complicate the deployment of AI models. This PhD research focuses on the development of adaptive and energy-efficient AI models tailored for the IECC. Key contributions include automatic scaling of AI architectures, optimization of distributed AI systems and latency constraints, and validation of the created method through real world experiments.**

***Index Terms*—Iot, Edge, Cloud, AI, Energy efficiency.**

## I. Introduction

Low-power IoT sensors have seen significant development over the past several years. Their versatility allows them to be employed in numerous scenarios where mains-powered sensors are not usable. However, this comes at the cost of limited computational capabilities. Traditionally, the cloud has been used to address this limitation. This approach has enabled the development of numerous applications and remains widely adopted today. However, relying on the cloud introduces substantial latency while increasing communication cost and compromising data privacy. In response to these limitations, edge computing was introduced as an intermediate layer offering greater computational power than IoT nodes while being closer to sensors than Cloud servers.

The IoT-Edge-Cloud Continuum (IECC) aims to leverage these layers and create applications able to take full advantage of their respective strengths. This continuum represent an efficient solution for use cases such as health wearable, Augmented/Virtual Reality (AR/VR) or smart cities. These applications indeed need fast response time combined with excellent performance and reduced energy consumption. Therefore, understanding and optimizing the use of the IECC will be crucial in the future for these cutting edge applications. While there are undeniable advantages, the development of this continuum has faced many challenge:

This work has been supported by the French government, through the France 2030 investment plan managed by the Agence Nationale de la Recherche, as part of the "UCA DS4H" project, reference ANR-17-EURE-0004

- Heterogeneity : The devices characteristics and constraints, as well as their number, are different in the continuum, which makes it nearly impossible to find optimal solutions to optimization problems.
- Dynamic environment : While the topology of the continuum is known, its state at a given time can be impacted by a multitude of parameters : battery state of charge, computational load, state of the bandwidth and more.
- Multi-objective optimization : While many Quality of Services (QoS) can be optimized using IECC, managing several of them is challenging.

These problems are well known and subject to numerous studies in the literature. However, in most of these works, tasks are described as generic IoT tasks that can easily be distributed. While effective in some scenarios, distributing IoT tasks cannot be applied for AI inference tasks.
This PhD focuses on developing energy efficient AI models for the IECC taking advantage of its full capabilities. The following points will be thoroughly studied and lead to the main contributions of the PhD :

- The automatic and adaptive scaling of AI architectures on the target device, based on the resources available, while aiming to find the right balance between performance (e.g., accuracy, precision), resource consumption, and service requirements.
- The optimization of AI architectures for distributed intelligence across the IoT-Edge-Cloud Continuum (IECC), while preserving the data confidentiality. To minimize communication latency, this optimization must be carried out in conjunction with an efficient allocation of computational and communication resources.

## II. Related works

To address these challenges, we reviewed the different solution currently proposed by the literature. Three major approaches were identified to distribute deep learning models across a large number of devices.

### A. Ensemble models

Ensemble models seek to create prediction models expert in a particular subgroup of all classes. An aggregation of all models prediction is then done through different processes. This method yields excellent results in terms of accuracy but

leads to higher computational load, both at training and inference time. Nevertheless, ensemble model seems a promising solution for the IECC. In [1], for instance, the authors have successfully deployed an ensemble of small models suited for IoT nodes (i.e. nodes that respect power consumption constraints) while reaching the accuracy of bigger neural network .

### *B. AI Splitting*

AI splitting aims to separate a deep learning network in sub part that can then be executed on different machines. The different existing approaches in the literature can be classified in two categories. First, layer wise AI splitting type of approach splits the model after certain layers. Doing so allows to reduce the cost of data transmission, the output of an intermediate layer being often way smaller than the initial data. Introduced first in [2], many variants of this method have been proposed, like introducing a bottleneck [3], creating multiple splitting point [4] or using early exit [5].
On the other hand, parallelized splitting aims to create multiple independent partitions. Doing so allows some degrees of parallelization, thus greatly decreasing the models execution time. While this method results in smaller latency, its complexity and negative impact on accuracy usually makes it hard to implement. Some of the most notable works on parallelized splitting can be found in [6] and [7].

### *C. Neural Architecture Search*

Neural architecture search (NAS) is used for a wide range of neural networks. Using NAS-based methods, optimal architecture can be search and found, leading to better performance for a desired task. NAS has already been successfully applied to AI splitting in [8] where a convolutional neural network (CNN) with integrated bottleneck was proposed, allowing for better performance.
This approach can also be useful to create models suited for the target device, reducing the computational load while maintaining good performance.

## III. Ongoing work

We are currently working on a new parallelized splitting method based on [6]. In this paper, the authors present a two-step training to create uniform semantic groups inside layers that would allow model parallelization and parameter reduction. This can be used to distribute an AI model over the continuum and speed up its inference time. We focus on improving this approach by addressing one of its shortcomings. Having uniform group size is sufficient when multiple GPUs are used for inference, but it is not suited for scenarios with heterogeneous nodes having different computing capabilities. Therefore, the regularization parameters must be changed so that the size of each group can be chosen before the execution. By doing so, group size will be adapted to the target device, allowing for an optimal use of the resources on the continuum. Figure 1 shows the workflow of our proposed solution.

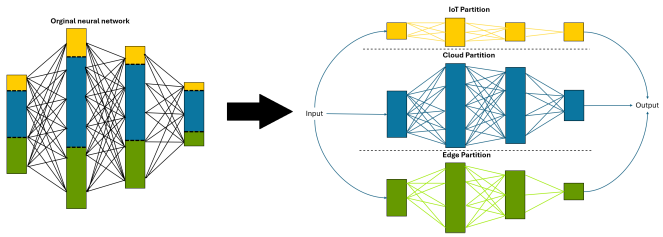



Fig. 1. Non uniform parallelized splitting for the IECC

## IV. Conclusion

This PhD aims at providing efficient solutions for the IECC to execute deep learning models with low energy consumption and low latency. In the future, we will :

- Propose approaches to distribute AI over the continuum
- Collect data from IoT sensors, Edge and Cloud nodes to create a realistic test environment
- Compare our methods with the state of the art
- Validate our solutions by deploying in a real world environment

## References


[1] Oihane Gómez-Carmona, Diego Casado-Mansilla, Diego López-de Ipiña, and Javier García-Zubia. Optimizing computational resources for edge intelligence through model cascade strategies. *IEEE Internet of Things Journal*, 9(10):7404–7417, 2022.

[2] Kang et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *ASPLOS '17*, ASPLOS '17, page 615–629, New York, NY, USA, 2017. Association for Computing Machinery.

[3] Jiawei Shao and Jun Zhang. Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2020.

[4] Haneul Ko, Bokyeong Kim, Yumi Kim, and Sangheon Pack. Two-phase split computing framework in edge–cloud continuum. *IEEE Internet of Things Journal*, 11(12):21741–21749, 2024.

[5] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 328–339, 2017.

[6] Juyong Kim, Yookoon Park, Gunhee Kim, and Sung Ju Hwang. SplitNet: Learning to semantically split deep networks for parameter reduction and model parallelization. In *ICML'17*, volume 70 of *Proceedings of Machine Learning Research*, pages 1866–1874. PMLR, 06–11 Aug 2017.

[7] Jiale Chen, Duc Van Le, Rui Tan, and Daren Ho. Nnfacet: Splitting neural network for concurrent smart sensors. *IEEE Transactions on Mobile Computing*, 23(2):1627–1640, 2024.

[8] Shoma Shimizu, Takayuki Nishioy, Shota Saito, Yoichi Hirose, Chen Yen-Hsiu, and Shinichi Shirakawa. Neural architecture search for improving latency-accuracy trade-off in split computing. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1864–1870, 2022.

# A retrospective on DISPEED – Leveraging heterogeneity in a drone swarm for IDS execution

Vincent Lannurien*, Camélia Slimani*, Louis Morge-Rollet*,
Laurent Lemarchand†, David Espes†, Frédéric Le Roy*, Jalil Boukhobza*
Lab-STICC, CNRS, UMR 6285, { * ENSTA, Institut Polytechnique de Paris, † Université de Bretagne Occidentale }, Brest
Email: {vincent.lannurien, frederic.leroy, jalil.boukhobza}@ensta.fr,
camelia.slimani@toulouse-inp.fr, louis.morge-rollet@grenoble-inp.fr, laurent.lemarchand@univ-brest.fr

***Abstract*—Swarms of drones are gaining more and more autonomy and efficiency during their missions. However, security threats can disrupt their missions' progression. To overcome this problem, Network Intrusion Detection Systems ((N)IDS) are promising solutions to detect malicious behavior on network traffic. However, modern NIDS rely on resource-hungry machine learning techniques, that can be difficult to deploy on a swarm of drones. The goal of the DISPEED project is to leverage the heterogeneity (execution platforms, memory) of the drones composing a swarm to deploy NIDS. It is decomposed in two phases: (1) a characterization phase that consists in characterizing various IDS implementations on diverse embedded platforms, and (2) an IDS implementation mapping phase that seeks to develop selection strategies to choose the most relevant NIDS depending on the context. On the one hand, the characterization phase allowed us to identify 36 relevant IDS implementations on three different embedded platforms: a Raspberry Pi 4B, a Jetson Xavier, and a Pynq-Z2. On the other hand, the IDS implementation mapping phase allowed us to design both standalone and distributed strategies to choose the best NIDSs to deploy depending on the context. The results of the project have led to three publications in international conferences, and one publication in a journal.**

***Index Terms*—Swarm of drones, Network Intrusion Detection Systems, heterogeneous computing**

## I. Introduction

Unmanned Surface Vehicles (USVs) are used to carry out large-scale missions, which can involve drones with high computing power and autonomy, as well as less expensive drones with limited computing and battery capacity.

USVs operating in swarms (cooperation between USVs) can accomplish far more complex missions. However, this implies a high level of communication between drones, which can expose them to a variety of attacks. It is therefore necessary to detect attempted intrusions in good time and at low energy cost, using Intrusion Detection Systems (IDS).

Modern IDSs are mostly based on machine learning (ML) methods, where models are trained to identify abnormal and potentially malicious traffic. However, running inferences on ML models is generally costly in terms of computing and storage resources (main memory and disks), as well as consuming energy. It is therefore essential to optimize their execution so that the IDS task has the least impact on resource use, for USVs to accomplish their main tasks while ensuring a satisfactory level of safety.

One possible approach is to take advantage of the hardware heterogeneity that can exist among USVs, to choose the hardware configurations best suited both to the level of security required according to the geographical area in which the swarm is evolving, and to the resources available (computing and memory capacity) on the USV.

AID: Agence de l'Innovation de Défense

## II. Overview of the DISPEED project

The aim of DISPEED is to explore possible trade-offs in terms of safety, performance and energy for executing IDS on a swarm of USVs by exploiting intra/inter-USV hardware heterogeneity. Figure 1 shows the general operation of the platform designed as part of the project:

- ❶ **Characterization of IDS models and execution platforms**: IDSs are based on different machine learning algorithms (e.g. random forests, DNN) and deployed on heterogeneous computing elements (e.g. CPU, GPU, FPGA). We propose an offline methodology to characterize this environment in terms of Quality of Service (QoS) – latency, accuracy, energy consumption – and resource metrics (memory usage, storage, etc.) [1], [2]. A key finding of this study is the significant disparity in implementations characteristics, with many metrics exhibiting trade-offs (e.g., higher accuracy often comes at the cost of increased latency or energy consumption). Consequently, the optimal IDS implementation depends on the specific constraints of a given mission, motivating further investigation in studies ❷ and ❸;
- ❷ **Distribution of traffic to be analyzed within the swarm**: based on measurements from the offline phase, an online optimization strategy is implemented to decide on a distribution of the traffic to be analyzed between the drones in the swarm, so as to make the best compromise between energy and QoS [3]. The distribution module is decomposed in two steps: (1) drone capacity self-assessment, and (2) flow distribution. On the one hand, the drone capacity self-assessment allows each drone to estimate its processing capacity, as well as its workload, and broadcast it to the rest of the swarm. On the other hand, the flow distribution allows determining how the

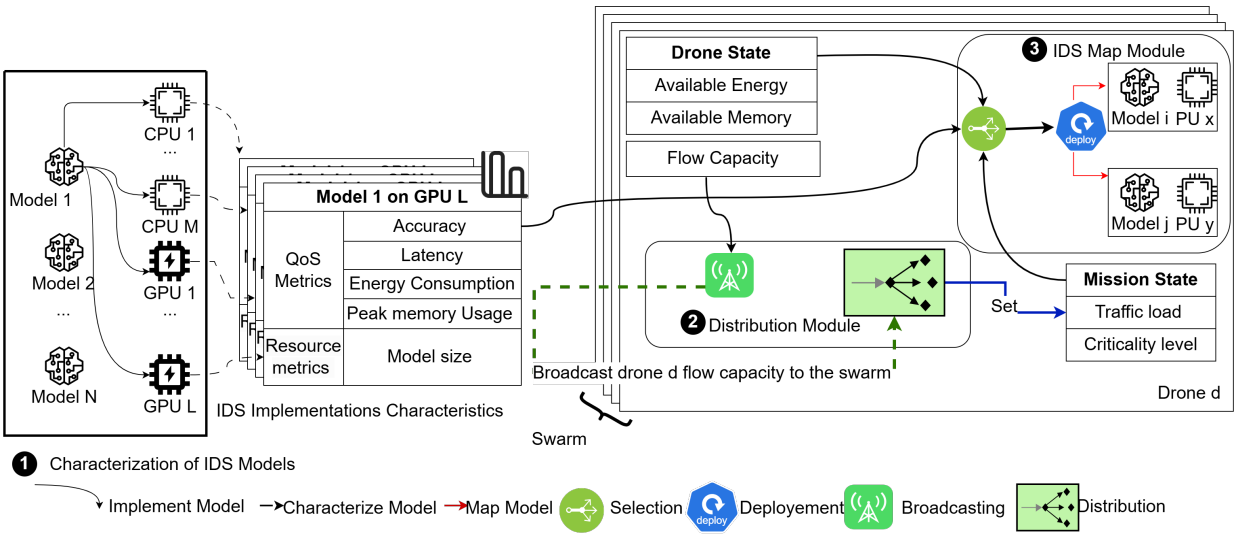



Fig. 1. High-level overview of the final system considered in DISPEED.

swarm will process the packet flows, while trying to minimize communications overhead. At the end of this phase, each drone can estimate its traffic load for the future phase;

- ❸ **Mapping IDSs to USVs**: the measurements from the characterization phase are used as part of a mixed offline/online optimization strategy that, depending on the state of each USV in the swarm, the characteristics of the IDS models and the state of the mission, selects the IDS model best suited to the situation and deploys it on the USV [4]. The selection process is divided in two phases: (1) the offline phase, which objective is to select implementations from the characterized set that lie on the Pareto front, while ensuring they meet the USV's storage constraints; and (2) the online phase, which consists in filtering the implementations that satisfy the live mission constraints, and choosing among them the implementation that hits the best trade-off between QoS metrics.

The aim is to explore different models of IDS, and implement them on different hardware architectures, in order to extract data on the level of security guaranteed by the model, as well as the time and energy cost of its implementations. Given the characteristics of all the implementations, the aim is to select the best implementation during the mission, *i.e.* the one that achieves the best compromise between security, performance and energy.

This heterogeneity can also be exploited to distribute the load in the swarm. The availability of each drone in the swarm can change during the course of a mission - depending on the criticality of the tasks allocated to it, but also in relation to a possible breakdown, for example. In this way, the distribution of the analysis work to be carried out on each drone can evolve during the course of a mission, so as to guarantee QoS. This flow distribution in a distributed system like a swarm of drones is not trivial, and requires rigorous analysis and optimization techniques to guarantee intrusion detection within the allotted time.

## III. Conclusion

In DISPEED, we devised a framework for IDS deployment on a swarm of heterogeneous drones. In order to optimize the system for energy consumption while enforcing QoS under security constraints, our solution considers the system at the granularity of the swarm to optimize traffic distribution across the drones; and at the granularity of a drone to optimize IDS selection. To sum up our general approach, DISPEED leverages hardware heterogeneity across a swarm of edge devices to satisfy resource constraints as well as operational constraints during various missions. Perspectives for future work include considering the actual deployment phase. Indeed, drones are mixed-criticality systems hosting workloads that compete for shared resources. As interferences between various processes arise on such capacity-limited devices, a scheduling strategy that consider individual drones as well as the swarm as a whole might be necessary to maintain adequate levels of QoS.

## References


[1] C. Slimani, L. Morge-Rollet, L. Lemarchand, F. Le Roy, D. Espes, and J. Boukhobza, "Characterizing intrusion detection systems on heterogeneous embedded platforms," in *2023 26th Euromicro Conference on Digital System Design (DSD)*, Durres, Albania, Sep. 2023, pp. 278–285.

[2] C. Slimani, L. Morge-Rollet, L. Lemarchand, D. Espes, F. Le Roy, and J. Boukhobza, "A study on characterizing energy, latency and security for intrusion detection systems on heterogeneous embedded platforms," *Future Generation Computer Systems*, vol. 162, p. 107473, 2025.

[3] L. Morge-Rollet, C. Slimani, L. Lemarchand, D. Espes, F. Le Roy, and J. Boukhobza, "DisPEED: Distributing Packet flow analysis in swarm of heterogeneous EmbEddeD platforms," in *2025 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2025.

[4] L. Morge-Rollet, C. Slimani, L. Lemarchand, F. L. Roy, D. Espes, and J. Boukhobza, "IDS-DEEP: a strategy for selecting the best IDS for Drones with heterogeneous EmbEdded Platforms," in *2024 IEEE 36th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2024, pp. 138–147.

# Comparative Study of Safety and Security-Protected AES Designs

Daniel THIRION[†§], Jean-Marc DAVEAU[†], Valentin EGLOFF[§],
David HELY[§], Vincent BEROULLE[§], Philippe ROCHE[†]

[†]STMicroelectronics, 850 Rue Jean Monnet 38926 Crolles Cedex, France (*firstname.lastname@st.com*)

[§]Univ. Grenoble Alpes, LCIS, 50 Rue Barthélémy de Laffemas, 26000 Valence, France (*firstname.lastname@lcis.grenoble-inp.fr*)

***Abstract*—With the rise of cybersecurity requirements in critical systems like vehicles and satellites, combining functional safety and hardware security is essential. While safety and security methods are studied individually, their combined analysis at the RTL or Netlist level is less explored. This paper analyzes multiple AES designs—unprotected, safety-protected (Lockstep), and security-protected (Parity-Predictor)—using simulation and formal methods. We highlight challenges and opportunities for combined assessments, evaluate designs against ISO 26262 safety metrics, and analyze their resilience to laser fault attacks, providing insights into their security robustness and the interaction of safety and security.**

***Index Terms*—Functional Safety, Security, RTL Design, AES, Fault Attack, ISO 26262, Single-Event Upset, Formal Proof**

## I. Introduction

Functional Safety and embedded security are increasingly mandated for critical integrated circuits in aerospace or automotive. However, safety and security are often handled separately, overlooking potential interactions: safety countermeasures (e.g., redundancy) might increase the attack surface, while security measures (e.g., encryption) can complicate safety verification [1]. Early design-phase analysis of these interactions is crucial for reducing costs and time-to-market.

The literature lacks detailed studies on safety/security interactions at the RTL/Netlist level [2]. This work explores this gap by proposing an analysis workflow combining simulation and formal methods. We apply this workflow to compare an AES design protected for safety (Lockstep) against one protected for security (Parity/Predictor). Our goal is to answer: how does a safety-protected design fare against security threats, and vice versa? Which tools and methods are suitable for this combined analysis at the hardware design level?

This paper first presents the safety and security analysis methodologies and tools. Then, it applies them to the AES case study, comparing the effectiveness of safety and security countermeasures across both domains. Finally, it discusses the results and methodological insights.

## II. Methodology

### A. Safety Analysis

Functional safety aims to ensure safe operation even with hardware faults, primarily targeting random radiation-induced Single Event Upsets (SEUs) and Transients (SETs) [3]. Countermeasures often involve redundancy. We assess effectiveness using metrics and fault classes defined in ISO 26262. Faults are categorized based on output correctness and detection status ($2 \times 2$ matrix of Unobserved/Dangerous + Undetected/Detected), and classes are defined as:

- $\lambda_{SPF}$: Single Point Faults (Dangerous Undetected [DU] in unprotected area)
- $\lambda_{RF}$: Residual Faults (DU in protected area)
- $\lambda_{MPF/L}$: Latent Multiple Point Faults (DU when fault injected in both functional and countermeasure)
- $\lambda_{MPF/D}$: Detected Multiple Point Faults (DD)
- $\lambda_S$: Safe Faults (UU, DU or DD)

Key metrics derived are Diagnostic Coverage ($DC$), Single-point fault metric ($M_{SPFM}$), and Latent fault metric ($M_{LFM}$). Higher values are better. We use Monte-Carlo fault simulation to inject random SEUs and estimate these metrics [4].

### B. Security Analysis

Hardware security analysis often follows frameworks like Common Criteria or ISO/SAE 21434. We focus on fault attacks (e.g., laser, glitching), which intentionally induce faults to extract secrets. Unlike safety's random fault model, security assumes a targeted attacker capable of inducing potentially complex, multi-bit faults. As such, this analysis requires exhaustive vulnerability search.

Our workflow involves: (1) Understanding the design and known attacks (e.g., Differential Fault Analysis on AES). (2) Defining security properties (e.g., "valid ciphertext or error raised"). (3) Using Formal Fault Injection (FFI) [5] to exhaustively find faults (locations and timing) that violate properties. (4) Analyzing found vulnerabilities for exploitability.

Instrumentation of the design flip-flops is done to allow the formal engine to inject faults (bit-flips, SEU model) and find counterexamples. Unlike simulation, FFI aims for full proof or finding all violations under given constraints.

## III. AES Case Study

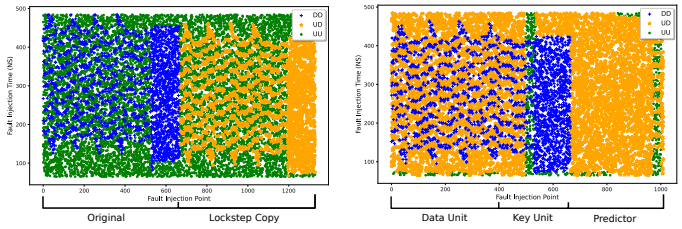
We apply our methodology to three AES-128 designs:

- **Vanilla**: Base design from [6] (countermeasures removed), serving as a reference. Consists of data, key, and control units.
- **Lockstep**: Duplicates the vanilla AES with a 6-cycle delay between cores . Outputs are compared; mismatches raise an error. This is a typical safety countermeasure.
- **Parity**: Design from [6] using parity prediction . Parity bits are added to data/key paths and checked against predicted values. Designed against laser fault attacks. Control unit is unprotected per original source.

Error handling (e.g., recomputation, masking output) is assumed at the system level but not implemented here.

### A. Naive Fault Injection Comparison

Simulated single-bit fault injections (*Figs. 1*) provide a simple view of the design response to injection. The Lockstep design has a high fault masking (UU), reducing false positive rate (UD). The Parity design, however, has a high detection rate (UD), minimizing masked faults. This highlights a fundamental difference: safety often benefits from masking non-critical faults (minimizing false-positives), while security prefers detecting any deviation (maximizing detection).




(a) Lockstep (b) Parity

Fig. 1: Naive fault injection classification vs. injection time (Y-axis) and element index (X-axis)

### B. Safety Metrics Analysis

ISO 26262 metrics were computed via fault simulation (*TABLE I*). Lockstep achieves 100% $M_{SPFM}$, being fully resilient to single-point faults. Parity's $M_{SPFM}$ is slightly lower (98.04%) due to the unprotected control unit contributing to $\lambda_{SPF}$; if excluded or protected, it would also reach 100%. Both designs show high resistance to latent multi-point faults ($M_{LFM} > 99.9\%$), with Lockstep being marginally better due to the temporal separation making simultaneous identical errors harder. The Parity design, despite being security-oriented, demonstrates excellent safety performance against the ISO 26262 random fault model, meeting requirements potentially up to ASIL D (if unprotected parts addressed).

| | **Vanilla** | **Lockstep** | **Parity** |
|---|---|---|---|
| Instrumented FFs | 848 | 1696 | 1010 |
| $\lambda_s$ (%) | 70.38 | 85.32 | 80.59 |
| $\lambda_{SPF}$ (%) | 14.94 | 0.00 | 1.02 |
| $\lambda_{RF}$ (%) | N/A** | 0.00 | 0.00 |
| $\lambda_{MPF/D}$ (%) | N/A** | 14.68 | 18.35 |
| $\lambda_{MPF/L}$ (%) | 14.68 | 0.01 | 0.04 |
| $DC$ (%) | N/A** | 100.00 | 100.00 |
| $M_{SPFM}$ (%) | 70.13 | 100.00 | 98.04 |
| $M_{LFM}$ (%) | 70.57($\pm$0.49*) | 99.996($\pm$0.01*) | 99.916($\pm$0.03*) |

*95% confidence level; **No countermeasure

TABLE I: Safety Metrics Comparison

### C. Security Analysis via Formal Methods

We use FFI to assess resilience against DFA, targeting the property: *"The ciphertext is valid OR the detection signal is raised."*. To simplify proof, we use a fixed key/plaintext. For Parity, we constrained FFI to ignore simultaneous injections into the same parity group (assumed uncovered by the protection).

Both Lockstep and Parity designs are proven secure against single-bit faults. For dual-bit faults (M=2), FFI found violating sets:

- *Lockstep*: 82 sets found, faults which are hard to explain due to the time difference and the complexity of AES.
- *Parity*: 2405 sets found (excluding same-group injections), exploiting faults in corresponding bits of the main logic and the predictor logic.

These M=2 vulnerabilities could potentially be exploited by precise laser attacks but might be mitigated by careful physical placement (separating Lockstep cores or Parity and Predictor cores). Formal analysis is significantly faster (2 years vs. 1 week) than exhaustive simulation for finding $M = 2$ sets. Formal proof struggled with $M = 3$ due to state space explosion and redundant findings of $M = 2$ sets.

## IV. Conclusion and Perspectives

This paper explored combined safety and security analysis of AES designs using simulation and formal methods. Our findings highlight key trade-offs:

- **Lockstep** (safety): high safety metrics ($M_{SPFM}$, $M_{LFM}$) and high fault masking, but its lower detection rate makes it less ideal for security.
- **Parity** (security): high safety metrics (ASIL D capable), could be a candidate in a safe and secure design. However, a high detection rate, while beneficial for security, creates many "false positives" from a safety perspective (detecting functionally safe faults).

This shows a need for countermeasures that can offer distinct error signals for safety-critical and security-critical events.

Methodologically, Formal Fault Injection is significantly more efficient than simulation for exhaustive security vulnerability searches, especially when the violation space is relatively small. However, its scalability to higher fault multiplicities ($M \geq 3$) remains a challenge.

Future work should explore other fault models (e.g., timing faults, SETs) and analyze non-cryptographic IPs to generalize these findings.

## References


[1] N. Wiersma and R. Pareja, "Safety != security: On the resilience of ASIL-d certified microcontrollers against fault injection attacks," in *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2017, pp. 9–16. DOI: 10.1109/FDTC.2017.15.

[2] Priyadarshini, S. Greiner, M. Massierer, and O.-E.-K. Aktouf, "Feature-based software architecture analysis to identify safety and security interactions," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, 2023, pp. 12–22. DOI: 10.1109/ICSA56044.2023.00010.

[3] G. R. Srinivasan, "Modeling the cosmic-ray-induced soft-error rate in integrated circuits: An overview," *IBM J. Res. Dev.*, vol. 40, no. 1, pp. 77–90, 1996. DOI: 10.1147/RD.401.0077.

[4] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, 2009, pp. 502–506. DOI: 10.1109/DATE.2009.5090716.

[5] D. Zuccala, J.-M. Daveau, P. Roche, and K. Morin-Allory, "Formal temporal characterization of register vulnerability in digital circuits," in *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jun. 2023, pp. 1–6. DOI: 10.1109/ISVLSI59464.2023.10238514.

[6] C. Ananiadis, A. Papadimitriou, D. Hély, V. Beroulle, P. Maistri, and R. Leveugle, "On the development of a new countermeasure based on a laser attack RTL fault model," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2016, pp. 445–450.

# Multi-Trojan Radio-Frequency Retroreflector Attacks

Pierre Granier, Marie-Aïnhoa Nicolas, Jordane Lorandel, Christophe Moy,
Philippe Besnier, Matthieu Davy, François Sarrazin
*Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, F-35000*
Rennes, France
{pierre.granier, marie.nicolas, jordane.lorandel, christophe.moy, matthieu.davy, francois.sarrazin}@univ-rennes.fr
philippe.besnier@insa-rennes.fr

***Abstract*—This paper demonstrate the feasibility of data exfiltration from multiple signal lines through the use of implanted retroreflectors, each tuned to distinct optimal interrogation frequencies. The effectiveness of the proposed attack is illustrated by recovering the color information of a picture transmitted over a VGA cable.**

***Index Terms*—Hardware Trojan, Electromagnetic information security, Backscattering, Intentional electromagnetic interference**

## I. Introduction

Information recovery from electromagnetic leakage is by now a well-known risk in electromagnetic cybersecurity. Extensive work on video display has been led since the pioneering work of Wim van Eck in 1985 [1] on analog CRT, later expanded on the VGA protocol [2], [3], and on TMDS based interfaces such as HDMI/DVI. Such attacks are classified as passive since they do not require any software or hardware modifications to the targeted device. Their success heavily depends on the target's electromagnetic emissions as well as the surrounding radio environment.

Beyond passive eavesdropping, an adversary capable of modifying a device (during the supply chain process or post distribution) can enable a broader class of attacks. Among these, hardware Trojans utilizing retroreflectors are particularly notable because they remain inactive until illuminated by an external RF source, making them more stealthy than permanent transmitters. The first known example of a retroreflector, called "The Thing" was used in the early days of the Cold War for eavesdropping surrounding audio. This type of attack, labeled as Radio-Frequency Retroreflector Attack (RFRA), was later adapted for spying on modern electronic as shown in leaked NSA documents [4] and later reproduced by academics [5].

This paper develops the concept of a multi-Trojan RFRA leveraging frequency diversity. This approach can target either multiple devices or multiple signal lines within a single target. To illustrate this concept, we showcase an attack on the three color components (Red, Blue, and Green) of a VGA cable, enabling the retrieval of colored images.

This publication is supported by the European Union through European Regional Development Fund (ERDF), Ministry of Higher Education and Research, CNRS, Brittany region, Conseils Départementaux d'Ille-et-Vilaine and Côtes d'Armor, Rennes Métropole, and Lannion Trégor Communauté, through the CPER Project CyMoCod, and by the French "Agence Nationale de la Recherche" (ANR) under Grant ANR-22-CPJ1-0070-01.

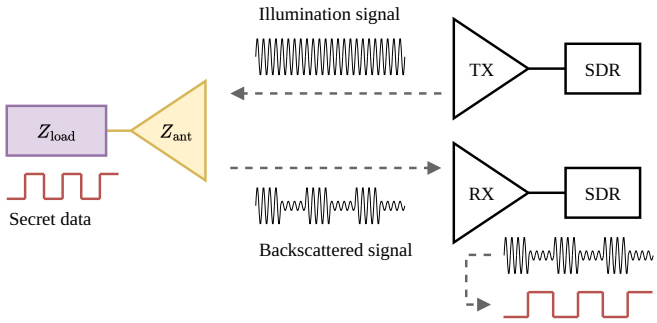



Fig. 1. RF Retroreflector Attack (RFRA) principle. It is composed of an SDR-based interrogation setup (right) and a target (left) consisting of an antenna terminated by a varying load.

## II. Trojan Architecture and Interrogation

The principle of RFRA is illustrated in Fig. 1. A hardware Trojan is embedded within a target, and connected to a signal line. The Trojan consists of an antenna whose load impedance varies in response to the data line signal. The changes at the antenna load impedance modify the reflective properties of the antenna, effectively making the Trojan a backscatter device (in a similar fashion to an RFID tag). Interrogation is made by illuminating the implant with a carrier wave and observing its reflection, both typically achieved using a software-defined radio (SDR).

In [4], [5] the Trojan's design is based on a transistor. The shielding of the targeted cable is cut, and the transistor's drain and source terminals are connected to the resulting separated shield sections, effectively forming a dipole antenna characterized by an impedance $Z_{\text{ant}}$. The signal line targeted for interception is connected to the transistor gate, driving the impedance $Z_{\text{load}}$ between drain and source. The reflection coefficient of the implant is given by $\Gamma = (Z_{\text{load}} - Z^*_{\text{ant}})/(Z_{\text{load}} + Z_{\text{ant}})$ dictating changes in the backscattered signal.

Our Trojan implementation (shown in Fig. 2) differs from the previously described architecture by two aspects. First, we switched from a transistor-based architecture to a diode-based architecture, applying a bias voltage across the anode and

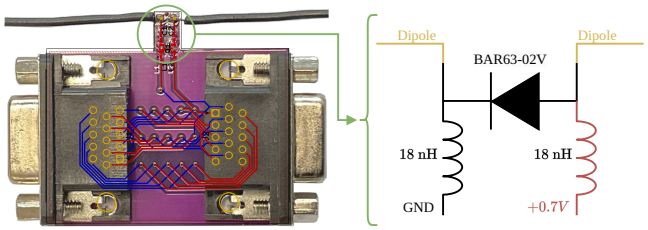



Fig. 2. VGA passthrough where a hardware Trojan probes the red signal line of the video data.

cathode of the diode, as opposed to the gate and source in the transistor-based configuration. Second, rather than repurposing the cable shielding, we chose to simplify the experiment by using custom-made PCBs similar to the work of [6] where the antenna part is made from added wiring creating a dipole. While this design is not optimized for stealth, it could be translated to a more covert proof of concept by introducing selective gaps in the cable shielding, enabling the formation of dipole antennas of various lengths.

## III. Multiple Trojan RFRA

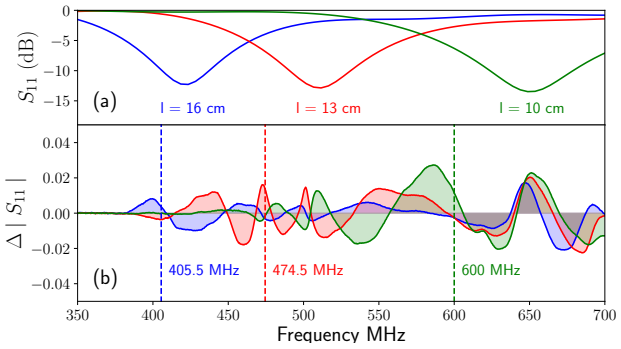



Fig. 3. (a) Simulation of the Trojans antennas $S_{11}$ (dB). (b) BAR63-02V based Trojan $\Delta|S_{11}|$ using three different antenna lengths. (Dashed lines) interrogation frequencies used for recovering the RGB matrix.

### A. Multi-trojan characteristic

We use the previously described Trojan, changing only the RGB line probed and the dipole length to obtain antennas of different resonant frequencies. The chosen antenna lengths are $l_{\mathrm{G}} = 10$ cm, $l_{\mathrm{R}} = 13$ cm and $l_{\mathrm{B}} = 16$ cm. Simulations of the reflection coefficients $S_{11}$ of the three antennas created are presented in Fig. 3(a), revealing resonant frequencies near 420 MHz, 520 MHz and 650 MHz for the blue, red and green Trojan, respectively.

For each Trojan, we can measure the difference of $|S_{11}|$ (shown Fig. 3(b)) using one of the interrogation antennas we will later use (a log-periodic LP0410 antenna) connected to a vector network analyzer. This difference in $|S_{11}|$, shown in Fig. 3(b), provides insight into the frequency-dependent effectiveness of each Trojan when amplitude demodulated.

### B. Experiment

Using the three previously characterized implants integrated into a VGA cable, we connect a laptop displaying the picture shown in Fig. 4(a) to an external desktop monitor. The laptop's graphics card transmits analog voltage levels between 0 and 0.7 V for each pixel's color channel (red, green, and blue), rendered sequentially pixel-by-pixel, line-by-line, and frame-by-frame.

Positioned approximately two meters away, we illuminate the target using a log-periodic antenna connected to a signal generator, while receiving the backscattered signal with a software-defined radio (SDR) connected to a second log-periodic antenna. Amplitude demodulation and illumination are performed at the frequencies where each color component exhibits a dominant Trojan response, as identified in Fig. 3(b). By capturing a few frames per Trojan (i.e., per color channel), we reconstruct the RGB image shown in Fig. 4(b), yielding a visual approximation of the original image in Fig. 4(a).

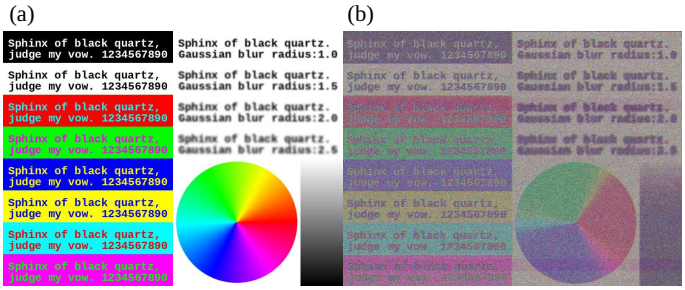



Fig. 4. (a) Targeted picture, (b) Rasterization result.

## IV. Conclusion

We demonstrated through the successful extraction of all three RGB channels that careful design in multiple Trojan implementations allow for frequency discrimination of their responses. This approach sparks the possibility of interrogating multiple co-located targets or distinct data lines within a composite link.

## References

[1] W. van Eck, "Electromagnetic radiation from video display units: An eavesdropping risk?" *Computers & Security*, vol. 4, no. 4, pp. 269–286, 1985, ISSN: 0167-4048.

[2] M. G. Kuhn, "Compromising emanations," in *Encyclopedia of Cryptography and Security*, 2003.

[3] M. Marinov, "Remote video eavesdropping using a software-defined radio platform," 2014.

[4] NSA. "NSA ANT catalog." (2008 (redaction)).

[5] M. Kinugawa, D. Fujimoto, and Y. Hayashi, "Electromagnetic information extortion from electronic devices using interceptor and its countermeasure," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 4, 62–90, Aug. 2019. DOI: 10.13154/tches.v2019.i4.62-90.

[6] M. Ossmann. "The NSA playset: RF retroreflectors." (2014).

# Accelerating Transformer Networks on FPGA

Two-page paper for GDR SoC$^2$ poster session submission

Eric Chen*† Hadi Saoud* Nicolas Vigne* Florent de Dinechin†

*cortAIx Labs, Thales Research & Technology, 91767 Palaiseau, France
†INSA Lyon, Inria, CITI, UR3720, 69621 Villeurbanne, France
eric.chen@thalesgroup.com

***Abstract*—In 2017, Google published an article proposing a neural network architecture based on the attention mechanism, named transformer. This new neural network has revolutionized the field of artificial intelligence (AI) by becoming the new state-of-the-art in various tasks. However, they have higher computational complexity and larger model sizes compared to recurrent neural networks (RNNs) or convolutional neural networks (CNNs), which leads to strong embedded system constraints. This work is a survey of the difficulties to address when implementing transformer accelerators on field programmable gate arrays (FPGAs), such as hardware architecture, memory constraints and non-linear function approximation.**

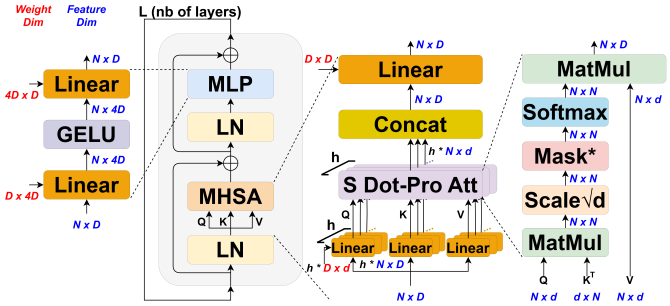
***Index Terms*—neural network, transformer, FPGA, computer architecture, high-performance computing.**

## I. Introduction

Transformers are a class of deep neural networks that have revolutionized AI by introducing a novel mechanism called attention [1]. This mechanism contextualizes data and ensures that relevant information is appropriately highlighted [1], [2], [3]. Transformers also rely on attention to capture long-range dependencies, which contrasts with CNNs that operate using localised convolution [2], [4]. However, this advantage increases the computational complexity of transformers. Most of today's accelerators adopt graphics processing units because of their high-performance computing and easy software implementation [1], [4], [5]. However, they are hardly compatible with embedded constraints like latency, power consumption or thermal dissipation [6], [7]. FPGAs are good candidates for transformers acceleration since they address many of these challenges, notably efficient performance and scalability [7], [8], [9]. Indeed, they deliver high performance with low latency, flexibility with their reconfigurable hardware, and efficiency with their low power consumption. This work will present the difficulties encountered when implementing transformer accelerators on an FPGA.

## II. Transformers

Transformers consist of multiple successive layers known as encoder/decoder, which are presented in Fig. 1b. Each layer is composed of sub-layers, featuring layer normalization (LN), residual connection, multi-head self attention (MHSA), and multilayer perceptron (MLP). MHSA is used for the attention while MLP is used to introduce non-linearity with an activation function called Gaussian Error Linear Unit (GELU).




(a) MLP (b) Encoder decoder (c) MHSA (d) Scaled dot-product attention

Fig. 1: Transformer encoder/decoder layer architecture.

To delve deeper into the sub-layers, MHSA is made of different operations, comprising linear operations, concatenation, and self dot-product attention along with a non-linear function referred to as Softmax (shown in Fig. 1c and Fig. 1d). These operations manipulate two-dimensional matrices of size up to $N \times D$. Table I describes parameters with example values from a vision transformer (ViT) [3], [5]. The "head" in MHSA refers to performing multiple initial linear operations and scaled dot-product attention by multiple parallel instances.

The MLP depicted in Fig. 1a contains two fully connected (FC) layers using linear operations, separated by the GELU non-linear function. These operations manipulate two-dimensional matrices of size up to $D \times 4D$.

As discussed, transformer encoder/decoder layers utilize multiple large-sized matrices in their operations, which increases the complexity of transformer accelerators on FPGAs. They also exploit non-linear functions like Softmax and GELU which are difficult to approximate in hardware with limited resources and without accuracy degradation.

TABLE I: Usual ViT parameters

| Parameters | Description | ViT [3], [5] value |
|---|---|---|
| $L$ | number of layers | 12 |
| $h$ | number of heads | 12 |
| $N$ | token dimension | 196 |
| $D$ | model dimension | 768 |
| $d$ | model dimension per head | $d = D/h = 64$ |

## III. Hardware implementation constraints

As presented in the previous section, transformers are composed of successive encoder/decoder layers, where each one includes various operations on matrices. Therefore, the main challenges are:

*A.* Implementing an efficient architecture.
*B.* Addressing memory constraints.
*C.* Approximating non-linear functions.

### *A. Efficient architecture*

In FPGAs, two main types of architecture are used to design an accelerator: the spatial architecture and the temporal architecture [7]. Other designs are variants or hybrids that incorporate the two primary architectures, and can be implemented at different scales from coarse to fine granularity.

The spatial or dataflow architecture consists of multiple engines where each engine can only perform one type of function. The output of an engine serves as the input of another, and data are processed in a pipelined manner. This type of architecture is very efficient in performance and resources but suffers from a lack of flexibility and customization. It also requires careful tuning to ensure consistent throughput, otherwise the accelerators will significantly slow down due to bottlenecks caused by the starve and stall of each engine [8].

The temporal or overlay architecture consists of a single engine that can perform multiple functions. This design offers the advantage of being flexible and micro-controllable, but only one input is processed at a time. It also suffers from resource consumption due to multiple intermediate buffers [9].

The main difficulty is to find an architecture that can ensures efficient inference with minimal impact on accuracy, while also providing high throughput, low latency, flexibility and limited resource utilization.

### *B. Memory constraints*

Transformers have a larger model size and memory footprint compared to RNNs or CNNs [9], [10], [11]. For example, ViT has over 86 million parameters [3]. The on-chip memory resources in FPGAs are highly performant but limited in size. In contrast, off-chip memory offers significantly larger capacity but is much slower and introduces higher latency [9].

The main challenge is to manage the multiple large matrices involved in transformers within the limited on-chip memory, while efficiently handling data movement from off-chip memory without increasing the latency.

Quantization is a method that can also help alleviate memory constraints by reducing the model's data to smaller bitwidths or less complex datatypes [6], [10]. It primarily targets the model's activations and weights to reduce memory bandwidth usage and computation but it can affect the accuracy, since fewer bits are allocated for processing. Usually, neural network models operate with floating-point precision but FPGAs are much more efficient with integer-only computation. This lead to hardware implementations using integer or fixed-point with bitwidths that are no higher than 16 bits [6], [7], [10], [12], [13].

To address the memory constraints, choosing the correct quantization is essential to reduce memory footprint and accelerate transformer models with minimal accuracy degradation.

### *C. Non-linear functions*

Transformers exploit non-hardware-friendly non-linear functions Softmax and GELU in each transformer block [2], [4], [3], [5]. Therefore, approximating these functions for acceleration is necessary to increase throughput and reduce latency. The difficulties come from the degradation in accuracy when approximating these functions in hardware under limited power and resource utilization. Several approximations have been proposed, including shift operations [10], changing the base exponential to base 2 and logarithms [12], polynomial second-order approximations [13], tabulating the value in LUT [13], [14] or completely changing the activation function to a more hardware-friendly one [9].

## IV. Conclusion

Designing an FPGA-based accelerator for transformers proves to be challenging due to numerous embedded constraints. Transformers have larger model sizes compared to RNNs or CNNs. They also have higher computational complexity because of multiple successive layers that use non-linear functions. This generates constraints when designing an accelerator such as hardware architecture, memory constraints and non-linear function approximations.

This paper explores these main challenges when implementing transformer accelerators on FPGAs.

## References


[1] A. Vaswani et al., "Attention is All you Need," in *NIPS*, 2017.
[2] Z. Liu et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *ICCV*, 2021.
[3] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021.
[4] W. Wang et al., "InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions," in *CVPR*, 2023.
[5] H. Touvron et al., "Training data-efficient image transformers & distillation through attention," in *ICML*, 2021.
[6] Z. Liu, G. Li, and J. Cheng, "Hardware Acceleration of Fully Quantized BERT for Efficient Natural Language Processing," in *DATE*, 2021.
[7] H. Chen et al., "Understanding the Potential of FPGA-Based Spatial Acceleration for Large Language Model Inference," in *ACM Transactions on Reconfigurable Technology and Systems*, vol. 18, no. 1, pp. 1-29, 2024.
[8] T. Wang et al., "ViA: A Novel Vision-Transformer Accelerator Based on FPGA," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4088-4099, 2022.
[9] K. Marino, P. Zhang and V. K. Prasanna, "ME- ViT: A Single-Load Memory-Efficient FPGA Accelerator for Vision Transformers," in *HiPC*, 2023.
[10] Z. Li and Q. Gu, "I-ViT: Integer-only Quantization for Efficient Vision Transformer Inference," in *ICCV*, 2023.
[11] J. Fowers et al., "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *ISCA*, 2018.
[12] M. Huang et al., "An Integer-Only and Group-Vector Systolic Accelerator for Efficiently Mapping Vision Transformer on Edge," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 12, pp. 5289-5301, 2023.
[13] Q. Dong, X. Xie and Z. Wang, "SWAT: An Efficient Swin Transformer Accelerator Based on FPGA," in *ASP-DAC*, 2024.
[14] T. Hubrecht, O. Desrentes and F. de Dinechin, "Activations in Low Precision with High Accuracy," 2024, hal-04776745. [Online]. Available: https://inria.hal.science/hal-04776745v1

# Exploiting Near-Data Processing for Far-Edge Applications

Fiona Santoro*†, Jean-Marc Philippe*, Philippe Coussy†, Kevin Martin†
*cortAIx Labs, Thales Research and Technology, Palaiseau, France
†Univ. Bretagne-Sud, CNRS UMR 6285, Lab-STICC, Lorient, France
fiona.santoro@thalesgroup.com

***Abstract*—Data movements between the levels of the memory hierarchy are a costly part of an application execution. They significantly increase the energy consumption and latency, while reducing the available bandwidth. A possible solution is to move the computation closer to the data, like in near-data processing and edge computing paradigms. In the latter, and especially for far-edge applications relying on ruggedized servers or embedded systems, the energy efficiency is paramount. This paper proposes to use near-data processing approaches and associated devices to optimize the performance and energy efficiency of far-edge servers. It presents typical applications that benefit from near-data processing paradigm and gives an analysis that provides first interesting properties that make applications good candidates for such an approach.**

## I. Introduction

Data movements are costly during an application execution. They represent a significant part of the total energy consumption and latency, and reduce the available bandwidth. For example, in cloud computing those transfers have raised in the past years due to the substantial increase in data generated and collected by the end users devices and sensors. In fact, those movements can cause up to 60% of the total energy consumption of an entire datacenter application [1]. An other example is the memory hierarchy of computing systems, depicted in Fig. 1. For instance, an 8-bit integer multiply operation is orders of magnitude less energy consuming than a DRAM read [2]. To reduce these transfers, computations can be offloaded closer to the data, such as in edge computing and near-data processing (NDP) approaches. More precisely, the edge computing paradigm consists in offloading parts of the applications to the end users devices or intermediate computing platforms between the end user and the cloud servers. Inside a computing server, near-data processing reduces internal data movements using Computational Storage (CS) or Processing in Memory (PiM) techniques. They respectively place compute resources at the storage or the main memory levels of the memory hierarchy, as illustrated in Fig. 1.

Devices implementing these techniques could be interesting components of far-edge computing architectures, which rely on ruggedized servers or embedded systems. These architectures can be found in drones, radars, military vehicles where performance and energy efficiency are of high interest. This paper proposes to use NDP paradigms to improve far-edge applications. It presents a first analysis on key characteristics of applications that make them suitable for applying NDP techniques in the far-edge computing context. The remainder of this work is organized as follows. Section II explains the paradigm of NDP and lists some related devices. Section III analyzes typical NDP applications to extract their key characteristics. Section IV concludes the paper.

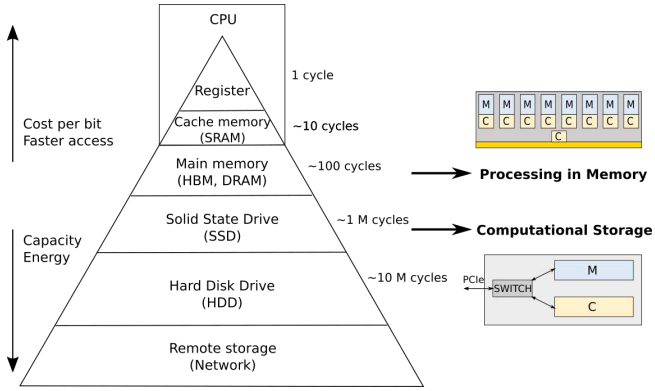



Fig. 1. Typical memory hierarchy diagram including near-data processing PiM and CS approaches with example architectures and latencies. M represents the memory technology (e.g. DDR for PiM and flash for CS) and C the added compute element (e.g. processor units for PiM and an FPGA for CS).

## II. Near-data processing paradigm

### A. Introduction of the NDP concepts

Modern computing systems implement a memory hierarchy to create the illusion of an infinite and fast memory for the processing unit (CPU). This hierarchy is based on several levels of different memory technologies with data transfers between them. At the top, there are the fastest memories but with reduced capacity and, at the bottom, those with the highest capacity and latency. NDP paradigm consists in placing a compute unit at one level of the memory hierarchy (see Fig. 1). In this study on far-edge systems, we only consider main memory and storage levels (PiM and CS).

**Processing in Memory** consists in adding compute capabilities at the main memory level (DRAM, HBM, etc.), as visible in Fig. 1. PiM can be subdivided into two main categories. The oldest one is **Processing Using Memory** (PUM), it uses intrinsic characteristics of the memory technology with minimal changes to implement new functionalities. The second one is **Processing Near Memory** (PNM), it adds a computing core near the memory. Industrially, it is usually near the banks.

In academical and theoretical approaches, cores are at the subarray and row-buffer levels. PiM efficiently reduces data transfers at the cost of a new programming model and code reorganisation to take advantage of it [3].

**Computational Storage** consists in adding computation capabilities at the storage level (Fig. 1). According to the Storage Networking Industry Association (SNIA), CS devices (CSx) can be defined as a storage component containing Computational Storage Engines (CSEs) (e.g. processors or FPGAs), aiming to execute Computational Storage Functions (CSFs) (specific storage operations such as compression). CSx can be classified into three categories. The first one is **Computational Storage Drive** (CSD), a storage element that contains one or more CSEs and persistent data storage (e.g. an SSD with internal compute capabilities). The second one is **Computational Storage Processor** (CSP), a component that contains one or more CSEs for an associated storage system, without providing persistent data storage itself (e.g. an FPGA board interacting with an SSD). The last one, **Computational Storage Array** (CSA), is a storage array with one or more CSEs (e.g. a RAID server with a CSA and multiple SSDs).

### *B. Related NDP devices*

This subsection presents examples of PiM and CS devices implementing the NDP approach.

**Processing in Memory.** UPMEM proposes a PNM device relying on custom DDR4 memory chips embedding general-purpose DRAM Processing Units (DPU). SK-Hynix introduces the AiM (Accelerator in Memory) GDDR6 chip including optimized hardware Artificial Intelligence (AI) computing elements (such as MAC or internal lookup tables). Samsung unveils AxDIMM (Accelerator DIMM), a DDR4-compatible FPGA-based PNM platform exploiting rank-level parallelism. It accelerates embedding lookup and pooling AI operations.

**Computational Storage.** AMD and Samsung proposed the SmartSSD, an U.2 drive in which an FPGA is connected to a 2TB NVMe flash disk thanks to PCIe switch. NGD Systems presented the NewPort platform with an ASIC accelerator, based on an ARM Cortex-A53 core embedded in the SSD controller. The ARM A53 core is also used in the Key-Value (KV)-CSD from Sk-Hynix and the Los Alamos National laboratory. Scaleflux proposes CSD 5000, an NVMe SSD with an online data compression at the write/read operations level.

## III. Analysis of typical NDP applications

After studying the state of the art regarding NDP, a significant part of the evaluation use cases can be roughly classified in four main categories. The first one is databases, like in [4] where NewPort is evaluated with MongoDB compared to SSDs. It reaches a performance of 3.4x faster and 57% less energy consumption in direct mode. The second category is Machine Learning/AI. In [5] the SmartSSD is added to the DeepSpeed ZeRO-Infinity technology, inside the storage offload of Large Language Models. This work shows an acceleration of up to 1.55x compared to SSDs. The third category contains all specifics applications such as genetics, radar, etc. For instance, in [6], the use of the UPMEM-PIM device results in a speed-up up to 157 for genetic mapping compared to a CPU+SSD system. The last category encompasses benchmarks used to evaluate some specific characteristics of a NDP device. These are not related to real world applications. For example, [3] characterizes some properties of UPMEM-PIM such as its compute throughput and memory bandwidth.

From the above studies, NDP is interesting to improve performances and energy efficiency on suitable applications, but it requires to rethink the way the application is designed. For example, applications that process large amounts of data (e.g. neural networks training, databases, etc.) increase the movements between the levels of the memory hierarchy. Typically, **memory-bound** applications are good candidates for PiM devices, whereas **storage-bound** applications can benefit from the CSx approach. The **characteristics** of the applications (e.g. operations, algorithm, data types and formats, access patterns, etc.), have also an impact on the choice of the NDP device. For example, UPMEM-PIM does not have native floating point operators contrary to AiM, but it can handle more application domains. Another example is the SmartSSD which is not optimized for small transfers due to the internal PCIe link. However, **software optimisation techniques** (at the application or system level) can be used to improve or enable the implementation on available hardware. Quantization or compression techniques like the CSR data format or the top-k approach are good examples of such optimisations.

## IV. Conclusion and perspectives

This paper proposes to explore the use of NDP paradigm in far-edge applications. A first analysis of related devices and applications benefiting from them, shows some key characteristics that were considered for introducing NDP approach in the state-of-the-art use cases. Generalizing data-centric profiling and software optimisation techniques can be interesting to efficiently exploit NDP in real world applications.

## References


[1] A. Boroumand et al., "Google workloads for consumer devices: Mitigating data movement bottlenecks," *SIGPLAN Not.*, vol. 53, no. 2, p. 316–331, Mar. 2018. [Online]. Available: https://doi.org/10.1145/3296957.3173177

[2] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2014, pp. 10–14.

[3] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," *IEEE Access*, vol. 10, pp. 52 565–52 608, 2022.

[4] A. HeydariGorji, S. Rezaei, M. Torabzadehkashi, H. Bobarshad, V. Alves, and P. H. Chou, "Leveraging Computational Storage for Power-Efficient Distributed Data Analytics," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 6, Oct. 2022, place: New York, NY, USA Publisher: Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/3528577

[5] H. Jang, J. Song, J. Jung, J. Park, Y. Kim, and J. Lee, "Smart-Infinity: Fast Large Language Model Training using Near-Storage Processing on a Real System," in *2024 IEEE International Symposium on High-Performance Computer Architecture*, Mar. 2024, pp. 345–360, iSSN: 2378-203X.

[6] D. Lavenier, C. Deltel, D. Furodet, and J.-F. Roy, "MAPPING on UPMEM," INRIA, Research Report RR-8923, Jun. 2016. [Online]. Available: https://hal.science/hal-01327511

# Design-Technology Co-Optimization Methodologies for a FeMFET Bitcell

Rosario Pronsato*, Ian O'Connor*, Pascal Vivet†
*CNRS, INSA Lyon, Ecole Centrale de Lyon, Universite Claude Bernard Lyon 1, CPE Lyon, INL, UMR5270, 69130 Ecully, France, CEA, List, F-38000 Grenoble, France
†LIST/DSCIN, CEA, Grenoble, France {firstname.lastname}@ec-lyon.fr, pascal.vivet@cea.fr

***Abstract*—Semiconductor technologies for computing hardware are made to pursue performance scaling through disaggregation (chiplets) of accelerator-rich architectures employing advanced paradigms such as approximate, vector and stochastic computing, as well as diversified integration of non-volatile memory, advanced 3D transistors, photonic and RF interconnect. The dimensionality of design space for computing hardware, from data centers to edge computing devices, is growing exponentially at a time when technology orientation is critical. To explore this vast design space, co-optimization techniques covering technology, circuits and systems and integrating the entire design value chain from technology models to application benchmarking are necessary. Such approaches require circuit and architecture design/synthesis and simulation tools, manufacturing process data, software aspects, compilers. The main objective of this work will be to enable the projection of technological developments of single emerging nanodevices on the design and performance of complex circuits and architectures for advanced applications. The work will initially be focused on the design and use of emerging non-volatile memory circuits in in-memory computing architectures for tensor processing and artificial intelligence hardware.**

***Index Terms*—Co-optimization, Emerging Technologies, Non-volatile Memories**

## I. Introduction

Common memory technologies used in traditional memory hierarchy are constrained by two main factors, latency and power. This represents a major bottleneck to improving performance as well as energy-efficiency in memory access dominated applications [1].

For these reasons present in processor-centric architectures, conventional computing systems cannot provide the required performance for modern devices and applications.

Alternatives being explored are memory-centric architectures. In this paradigm, some computational tasks are redirected into specialized memory units that are capable of processing data [2]. There are two main groups within these architectures: "Near-Memory computing" and "In-Memory computing". The former reduces the distance between the computing and the memory units, and the latter integrates processing in the memory unit, both thereby reducing the memory access time.

Emerging technologies for memory-centric computing -such as RRAM, Phase-Change Materials, FeMFET, among others- unlock new design opportunities, but introduce new parameters that lead to more complex trade-offs for the designer. Thus, it is necessary to employ co-optimization techniques that tackle technology, circuit and systems, integrating the entire design value chain from technology models to application benchmarking.

There is a wide range of such memory technologies emerging with a corresponding variety of benefits and drawbacks. This work will be focused firstly on FeMFETs, proposing a proper co-design and optimization at the bitcell level to then extend the approach to set of emerging technologies.

Recent work [3] highlight the capability of co-optimizing a full memory circuit, but this does not focus on the framework involved. Other studies related to iMC presents a system-level methodology to leverage the mapping efficiency [4] or the customizability of the architecture for better exploration [5].

In this work, we introduce the following contributions:

- a simulator-agnostic optimization framework that integrates heuristic algorithms with SPICE-like simulations, ensuring broad compatibility across different circuit design environments
- a case study of a 1T1C FeMFET bitcell with performance, power and area considerations at circuit level

## II. FeMFET Based Memory: Device Design and Analysis

### A. Device and Compact Model

In this section we briefly present the FeMFET device to then explain the subsequent stages of this project.

A FeMFET is designed by adding a ferroelectric (FE) layer to the gate stack of a MOSFET transistor as shown in Figure 1. The features of the ferroelectric material allow the device to have a non-volatile behaviour [6]. A sufficiently thick FE layer will give a low ratio of ferroelectric capacitance to gate dielectric capacitance. This way, the polarization of the FE layer can be retained, leading to a hysteric characterization of the output voltage as a function of the input voltage [7]. The application of a positive gate-to-source voltage ($V_{GS}$) beyond a threshold switches the polarization of the FE in the positive direction and the withdrawal of the said voltage does not affect the positive polarization achieved. On the same way, a sufficiently negative voltage $V_{GS}$ changes the state of the FE material and it is retained after the voltage goes back to zero.

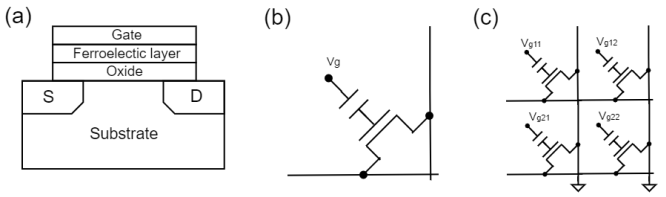



Fig. 1. (a) FeFET structure. (b) FeMFET bitcell. (c) 2x2 memory array.

### B. Design Technology Co-Optimization

Evolutionary algorithms have been used in research for decades. In principle, they are a stochastic search procedure with a population-based approach. This methodology has copied the biological mechanism of evolution, which means that it gathers its best population members, then mutates and recombines them from generation to generation to find an optimal design or until a certain criterion is met within specific constraints [8].

To perform co-optimization of the bitcell we used evolutionary optimization algorithms. The exploration of the FeMFET bitcell is formulated as a multi-objective optimization problem, due to the need to study different KPIs to be minimized or maximized.

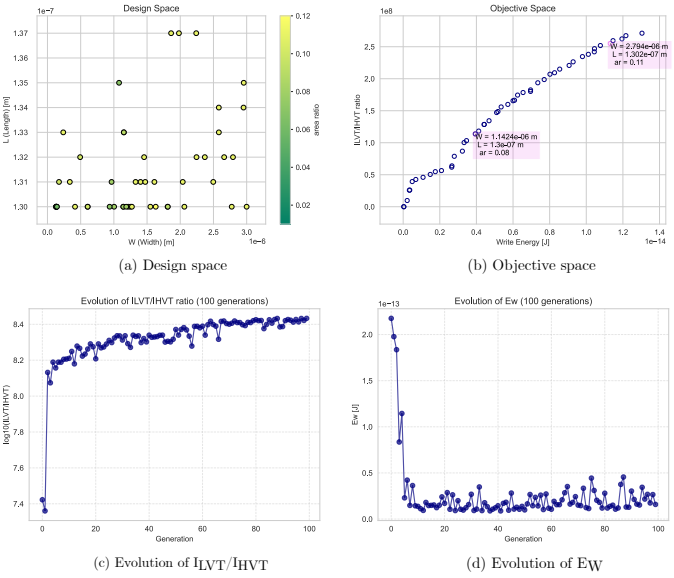
The optimization problem is made up of three parts: the definition of the variables in the design space, the objective functions, and the constraints. For our application, the variables in the design are shown in Table I.

TABLE I
DESIGN VARIABLES FOR OPTIMIZATION

| | | Unit | Boundaries | | | |
|---|---|---|---|---|---|---|
| | | | G01 | | G02 | |
| | | | Min | Max | Min | Max |
| Geometric Parameters | Length | m | 130n | – | 500n | – |
| | Width | m | 130n | – | 800n | – |
| | area ratio | – | 0.01 | 1 | 0.01 | 1 |
| Operating conditions | Programming voltage | V | 4 | | 4 | |
| | Reading voltage | V | 250m | | 250m | |

The Pareto-optimal solutions were generated using the Non-dominated Sorting Genetic Algorithm-II (NSGA-II), with 100 generations, a population size of 50, and 20 offspring per generation.

Figure 2(b) shows the Pareto front obtained from NSGA-II. It illustrates the tradeoff between maximizing the $I_{LVT}/I_{HVT}$ ratio and minimizing the write energy ($E_w$). As intended, the optimization would have been expected to rely mainly on the width of the transistor, the ferroelectric layer causing a more complex design space to work with. Opening more room for improvement with $A_r$ ranging between 2 to 12%. Additionally, sub-figures (c-d) represent the evolution of the best measured KPI of the generation. A high growth rate is observed between the first ten epochs but the one KPI reaching its maximum does not induce a straight extraction of the Pareto




(a) Design space (b) Objective space

(c) Evolution of $I_{LVT}/I_{HVT}$ (d) Evolution of $E_W$

Fig. 2. 2-objectives optimization between $I_{LVT}/I_{HVT}$ and $E_w$ varying $L$, $W$, and $A_r$.

front. Thus, solutions below the front are infeasible, while those above are suboptimal. This frontier being ideal and the one obtained experimentally resulting from the best elements of the population, it is necessary to run multiple epochs to converge to it. Break conditions offer the ability to stop the algorithm when no improvement is provided after a while.

## ACKNOWLEDGMENT

This work is founded by France 2030, in the context of the ANR PEPR électronique CHOOSE.

## REFERENCES

[1] W. A. Wulf and S. A. McKee, "Hitting the memory wall: implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, p. 20–24, mar 1995.

[2] H. A. D. Nguyen, J. Yu, M. A. Lebdeh, M. Taouil, S. Hamdioui, and F. Catthoor, "A classification of memory-centric computing," vol. 16, jan 2020.

[3] F. García-Redondo, L. Verschueren, S. Rao, P. Pandey, D. Abdi, P. Weckx, S. Couet, M. García-Bardon, and G. Hellings, "A novel dtco-driven 1t1r bitcell for sub-10ns stt-mram llc macros at n12 node," in *2024 IEEE European Solid-State Electronics Research Conference (ESSERC)*, pp. 665–668, 2024.

[4] K. Mishty and M. Sadi, "System and design technology co-optimization of sot-mram for high-performance ai accelerator memory system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 4, pp. 1065–1078, 2024.

[5] Y. Cai, Y. Gao, Z. Wang, L. Bao, L. Liang, Q. Zheng, C. Wang, and R. Huang, "Device-architecture co-optimization for rram-based in-memory computing," in *2023 IEEE 15th International Conference on ASIC (ASICON)*, pp. 1–4, 2023.

[6] A. I. Khan, C. W. Yeung, C. Hu, and S. Salahuddin, "Ferroelectric negative capacitance mosfet: Capacitance tuning antiferroelectric operation," in *2011 International Electron Devices Meeting*, pp. 11.3.1–11.3.4, 2011.

[7] X. Yin, A. Aziz, J. Nahas, S. Datta, S. Gupta, M. Niemier, and X. S. Hu, "Exploiting ferroelectric fets for low-power non-volatile logic-in-memory circuits," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, p. 1–8, IEEE Press, 2016.

[8] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary algorithms," *WIREs Data Mining and Knowledge Discovery*, vol. 4, p. 178–195, Apr. 2014.

# Un générateur de LFSR VHDL en sources ouvertes

Samuel Garcia [1]
[1]*Cedric - Laetita*
*CNAM*
Paris, France

***Résumé*—Les méthodologies de développement d'architecture électronique matérielle se heurtent à l'impossibilité du compromis performance/temps de développement. En effet, le recours à une architecture matérielle est le plus souvent dicté précisément par une forte contrainte de performance. Notre approche est de proposer des outils de génération de code VHDL permettant de rester sur une modélisation bas niveau avec un fort potentiel d'optimisation tout en s'appuyant sur un langage de scripte pour faciliter le développement du code et éviter d'avoir à réécrire des codes similaires à chaque projet pour quelques paramètres de différences. Nous présentons ici un tel générateur de code pour des architectures de génération de nombres pseudo-aléatoires de type LFSR.**

***Index Terms*—LFSR, random number generator, VHDL code generator, HDL methodology**

## I. Introduction

Le développement d'applications logicielles moderne repose majoritairement non pas sur la mise en œuvre algorithmique de concepts mathématiques avec des langages de programmation optimisés pour la performance, mais sur de petits scripts dont le rôle est d'assembler des briques logicielles préfabriquées. La récente tendance du "no code" propose même d'aller encore un pas plus loin dans cette direction. Le but n'est pas ici de produire le logiciel le plus performant, mais de répondre de façon rapide et efficace à une demande applicative.

De nombreuses solutions en termes de méthodologie et de logiciel de conception associés ont vu le jour dans les vingt dernières années pour donner au développement matériel une trajectoire similaire. Les plus notables sont la mise en place de bases de données de composants dits "sur étagère" et les produits de synthèse de haut niveau. Mais la pratique courante du développeur matériel moyen ne semble pas avoir été massivement influencée par ces propositions.

Pour comprendre pourquoi, il faut se rappeler pourquoi on a recours au développement d'architecture numérique matérielle sur mesure. Dans la très grande majorité des cas, c'est une contrainte de performance qui appelle cette approche. L'état d'esprit qui consiste à troquer de la performance pure contre de l'efficacité de développement est donc de facto beaucoup moins pertinent que dans le cas de développement d'applications logicielles.

D'un autre côté, la principale raison évoquée pour le non-recours à un développement matériel est précisément le manque d'efficacité du développement. Cela plaide pour la recherche de nouvelles méthodologies et/ou de nouveaux outils permettant l'amélioration de l'efficacité du développement matériel tout en représentant un compromis moins défavorable sur le plan des performances.

Pour répondre à cela, nous proposons des solutions basées sur des générateurs de code [2], [3]. L'idée centrale est de conserver au centre de la méthodologie l'utilisation de langages de description matériel bas niveau (e.g. : VHDL, Verilog, ...) qui permettent une maitrise fine sur l'architecture et donc un fort degré d'optimisation, mais d'automatiser une partie plus ou moins significative de la production de ce code final à travers des scrips de génération de code utilisant des langages de scripte spécialement adaptés à la manipulation de texte (e.g. : TCL, Python, ...).

Dans cet article, nous présentons un tel générateur de code spécialisé dans la génération de code VHDL de générateurs de nombres aléatoires de type LFSR.

## II. LFSR

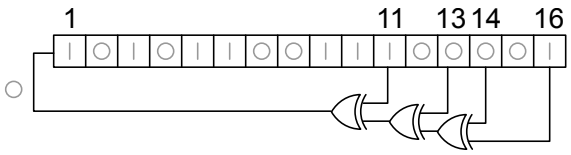



FIGURE 1. Architecture d'un LFSR dans sa version dite "Fibonacci"

De plus en plus d'applications requièrent l'utilisation d'un générateur de nombres aléatoires. Les LFSR représentent une méthode simple de génération de nombres aléatoires. Leur mise en œuvre est simple tant en logiciel qu'en matériel. S'ils sont loin de constituer des générateurs aléatoires parfaits, ils sont suffisamment bons pour la plupart des applications à l'exclusion des applications de cryptographie.

Les LFSR [1] [1] présentent plusieurs avantages, notamment leur simplicité matérielle, leur rapidité d'exécution et leur faible coût en ressources, ce qui les rend particulièrement adaptés aux applications embarquées et temps réel. Toutefois, en raison de leur nature déterministe et linéaire, ils peuvent présenter des vulnérabilités dans des contextes nécessitant un haut niveau de sécurité, comme le chiffrement de données sensibles. Pour pallier ces limites, des variantes plus complexes ou hybrides, combinant plusieurs LFSR ou les intégrant à des structures non linéaires, ont été proposées. Malgré ces déficits, les LFSR demeurent un outil fondamental

1. LFSR : Linear Feedback Shift Register ou registres à décalage à rétroaction linéaire

dans l'étude et la mise en œuvre de générateurs pseudo-aléatoires. Si un aléa plus poussé est nécessaire, un LFSR est fréquemment utilisé comme brouilleur pour améliorer encore la diversité d'un générateur physique. On a atteint alors un degré d'aléa suffisant pour les applications de cryptographie les plus critiques.

Les LFSR sont en cela une brique quasiment incontournable, et souvent suffisante, pour la génération de nombres aléatoires dans un circuit numérique.

La figure 1 montre l'architecture d'un LFSR. On y voit un registre à décalage circulaire dont la rétroaction provient de la combinaison de plusieurs bits de l'état courant à travers des opérateurs XOR. La séquence de nombres produite est périodique. Le choix de ces bits de rétroaction, qu'on appelle "taps", détermine la périodicité de la séquence. Plus la période est longue, meilleure est la qualité de l'aléa. Le choix des taps est donc un facteur déterminant pour la qualité du générateur aléatoire obtenue.

L'optimisation de ce choix a été longuement étudiée, et l'on trouve des tables [4] donnant, pour chaque longueur du registre à décalage, les taps à choisir pour maximiser la périodicité de la séquence. Le calcul de ces tables repose sur un algorithme non trivial, rendant très difficile l'intégration de ce calcul dans une description VHDL générique. C'est pourquoi l'utilisation d'un générateur de code qui va pouvoir facilement lire cette table dans un fichier fourni semble une approche plus pertinente.

## III. Le générateur de code

Le générateur de code proposé ici prend la forme d'un script Python appelé lfsrGen.py qui offre une interface utilisateur de type CLI typique des applications POSIX.

La page MAN de lfsrGen est donnée ci-dessous :

```
usage : lfsrGen [-h] [-n N] [-t TYPE]

VHDL Linear Feedback Shift Register
architecture generator for maximum
cycle pseudo-random scrambler

options :
  -h, --help  show this help message and exit.
  -n N        number of flip-flops in the
              register, default is 8
  -t TYPE, TYPE set the number of taps in
    --type    the lfsr, can be lfsr2 or lfsr4,
              default is lfsr2
```

Le choix des taps repose sur une table issue de [4] sous la forme d'un fichier CSV. Le fichier de table fourni contient les taps optimaux pour des LFSR de tailles de 2 à 256 pour une rétroaction à 2 ou 4 taps (selon les tailles, l'un ou l'autre peut ne pas exister). Cette table pourra être étendue si besoin.

Pour certaines tailles de registre, plusieurs jeux de taps offrent une périodicité similaire et maximale. Le but étant la recherche d'efficacité de développement, demander au développeur de faire un choix supplémentaire alors que cela n'aura pas d'impact significatif sur l'application ne semblait pas aller dans le bon sens. Il a donc été choisi de n'inclure qu'un seul jeu de taps par taille de registre et par type (2 ou 4 taps), choisi de manière purement arbitraire.

Il existe deux architecture classiques permetant la mise en oeuvre d'un LFSR, la méthode dite "Fibonacci" et la méthode dite "Galois". La version actuel du générateur repose sur la méthode Fibonacci.

Le scripte lfsrGen ainsi que la table des taps est mis à disposition de la communauté sous licence libre sur la plateforme Renater Sourcesup à l'adresse suivante : https ://sourcesup.renater.fr/projects/lfsrgen [5].

## IV. Résultat de mise en œuvre

Comme exemple d'usage, nous avons généré un LFSR de taille 10 bits à 4 taps. La commande exécutée est la suivante :

```
./lfsrGen -n 10 -t lfsr4
```

Cela produit le fichier "lfsr_10bits_4taps.vhd" qui contient la description du générateur aléatoire choisi et qui pourra être intégré dans n'importe quel projet VHDL.

La figure 2

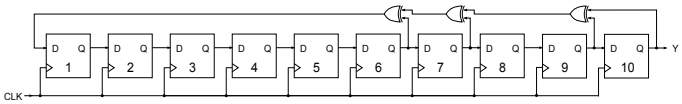



FIGURE 2. Architecture d'un LFSR 10 bits à 4 taps

Ce fichier a été mis en œuvre et testé avec succès sur un FPGA XC7Z020.

## V. Conclusion

Nous avons présenté un outil open source permettant la génération de code VHDL de générateurs de nombres pseudo-aléatoires de type LFSR. Il utilise une table des taps qui maximise la diversité et la périodicité de la séquence générée de 2 à 256 bits. Cet outil sera utile à quiconque a besoin de mettre en œuvre un générateur de nombres pseudo-aléatoires en VHDL. Les futures évolutions envisagées pour cet outil sont l'extension de la table pour des longueurs de registre jusqu'à 1024 et l'ajout d'une option pour générer une architecture de type "Galois". Une refonte est également envisagée pour utiliser le framework python vhdlGen en cours de développement.

## Références


[1] W.G. Chambers, "Clock-controlled shift registers in binary sequence generators", Computers and Digital Techniques, IEE Proceedings janvier 1988, p. 17-24.

[2] Garcia S., "Méthodologie de description matérielle d'ordre supérieur" 2019 - Colloque du GDR SoC2 2019

[3] Samuel Garcia, Ming-jun ZHANG, "Higher order HDL applied to MLP neural network hardware implementation" 2024 - 6th International Conference on Multidisciplinary Design Optimization & Applications (MDOA2024)

[4] Roy Ward, Timothy C.A. Molteno, "Table of Linear Feedback Shift Registers" 2012 – TECHNICAL REPORTS No. 2012-1 from the ELECTRONICS GROUP at the UNIVERSITY of OTAGO

[5] Dépôt du projet et des sources https ://source-sup.renater.fr/projects/lfsrgen

# FeMFET-Based Accelerator Design for iMC

Antoine Cauquil*, Damien Deleruyelle*, Ian O'Connor*
*Ecole Centrale de Lyon, INSA Lyon, CNRS, Universite Claude Bernard Lyon 1, CPE Lyon, INL, UMR5270, 69130 Ecully, France

***Abstract*—Non-volatile in-memory computing (iMC) has emerged as an energy-efficient paradigm well suited to AI workloads. Its implementation using 1T1C FeMFETs (Ferroelectric Memory Field Effect Transistors), a best-in-class emerging non-volatile memory technology that integrates BEOL ferroelectric devices with FEOL transistors, is of particular interest. This interest stems from their potential to enable large-scale multiply-accumulate (MAC) operations in both digital and analog domains targeting PTOPS/W efficiency.**

***Index Terms*—Ferroelectric, FeMFET, Bitcell, in-Memory-Computing, Design Space Exploration**

## I. Introduction

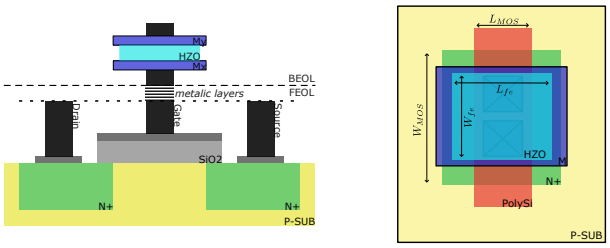
The rapid adoption of AI-driven applications, particularly large language models (LLMs) and edge AI, demands unprecedented computational power to handle increasingly complex models. However, physical miniaturization limits and the Von Neumann bottleneck create critical challenges. Shrinking transistor sizes faces fundamental physical barriers, limiting traditional hardware scaling for both datacenter LLMs and resource-constrained edge devices. Edge devices further struggle with thermal and power budgets, making conventional architectures inefficient for deploying billion-parameter models. Additionally, LLM inference requires moving vast amounts of data between memory and processors, consuming a significant portion of the energy due to redundant data transfers. For edge AI, this bottleneck exacerbates latency and power inefficiencies, hindering real-time applications such as autonomous robotics or on-device generative AI. To address these limitations, designers must explore innovative computing paradigms and leverage emerging technologies - such as ferroelectric materials - to fundamentally rearchitect AI hardware. Trends [1] show a shift to memory-centric specialized processing units, leveraging performances with data reuse and avoiding energy consuming fetch instructions [2]. Also known as the in-Memory Computing (iMC) paradigm, its goal is to use memory arrays to limit data movement and process multiple operations in parallel (multiply, accumulate and ReLU), based on its content and external stimulus.

This paper is structured as follows. In section II, we give an overview of ferroelectric materials and their capability for iMC. In section III, we describe the case study of a 1T1C FeMFET crossbar array and the challenges we face.

## II. Ferroelectric Bitcells

This part briefly introduces ferroelectric materials and discuss about different approach that can yield higher efficiency for AI inference.

Ferroelectric materials present compelling advantages for iMC by introducing non-volatility to conventional CMOS technologies. These materials exhibit a characteristic hysteretic polarization-field response. This study focuses on ferroelectric capacitors (FeCAPs) integrated in the back-end-of-line (BEOL), which retain polarization after voltage removal. By directly connecting the FeCAP to a MOSFET gate in a 1T1C FeMFET configuration as seen in figure 1, the remnant polarization modulates channel charge distribution, translating the material's hysteretic response into a shift of threshold voltage.




(a) FeMFET cross section (b) FeMFET layout

Fig. 1: FeMFET technology representation

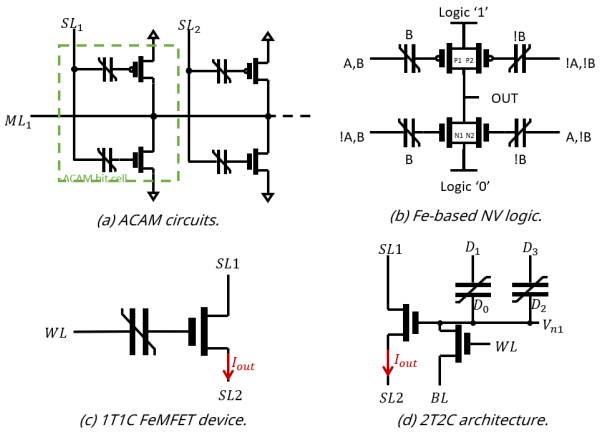



Fig. 2: Zoo of bitcells

FeFET technology unveiled new Content Addressable Memory architectures (fig.2a) with its NV capability that reduces the density of such circuits as low as 2T2C circuits. Although, its writing and reading can go beyond binary approach with analog programming of the Fecap. This Multi-Level Cell (MLC) behavior also reduces the density of search circuits by lowering the number of devices required. Nonetheless, it comes with higher constraints on accuracy as memory windows are smaller and more sensitive to noise or process variations. In term of AI inference, a circuit consisting of 2T2C has been demonstrated efficient [3] in the context of few shots learning and pattern recognition.

From a binary perspective, it is possible to replicate conventional logic circuits [4] [5] with a lower area. The Non-Volatility brought by the ferroelectric material can be

used to store operands that will be applied to an input vector avoiding fetch that are the main source of energy consumption [2]. The figure 2b draw an example with a XNOR logic cell. On a wider scope 2TnC structures exists where you can operate AND and OR operations in parallel [6]. Outside of these approaches, some architecture based on crossbar array of 1T1C FeMFET bitcell can execute vector and matrix multiplication in one clock cycle [7].

## III. Neuromorphic architecture

In this part we will focus on the FeMFET and its programming challenge.

### A. FeMFET crossbar array principle

Many bitcells can be organized within an array to act as a complete memory. Sharing the same drain and source by column; and gate by row, (see figure 3) it is possible to control them individually. To write data, a programming voltage ($\pm V_{prog}$) is applied on the row while the column signals are grounded to avoid drain-source conduction. For the reading part, a reading voltage ($V_{R_{ds}}$) is applied on a row while a small differential voltage is applied on the target column. With this principle, it is possible to process vector multiplication in one cycle between an input vector and the content of the array. The results are sensed as current per column.

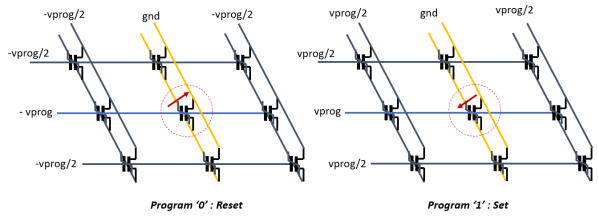



Fig. 3: Programming scheme for FeMFET crossbar array

### B. The programming challenge

Reference [8] highlights that the writing scheme presented in the previous section is not suitable for large arrays. In fact, when programming, the devices located in the half selected rows or columns, they see their values partially erased if not totally . Therefore, it can tear down the use of such approach for iMC. However, [8] proposes an approaches that mitigates these issues by applying half the programming voltage on the unselected rows and columns while programming. Nonetheless, experiments conducted in simulation with STMicroelectronic 130nm high speed CMOS technology and preisach based FeCAP [9] fitted with processed values from CEA-LETI still showcased disturb in the whole array. Initial FeMFET devices programmed with a remnant FeCAP polarity of -600mV (1) to -400mV for the fully unselected ones, (2) to +75mV for the half selected row, (3) to +250mV for the half selected column and (4) to +910mV for the targeted device to program. The variation of these voltages, shift the threshold voltage of the devices impacting the linearity of the MAC operations that are processed.

## IV. Conclusion

To fully address the programming issue that limits the scaling of the array for iMC application, future work will focus on exploring different programming scheme and optimizing them for best comparison. Prior concern will target solutions that offer reduced layout for the tiling and the footprint of the bitcells, which is the main advantage of the crossbar architecture. Other key performance indicators that will be used for optimization are linked to memory capability (endurance, retention time, peripheral circuit overhead) and computing performance (static power consumption, latency, cost per operation, linearity).


## Acknowledgment

We would like to express our sincere gratitude to CEA-LETI for providing real case values of ferroelectric capacitor. We also want to acknowledge Ferro4EdgeAI projects.


## References


[1] B. Murmann, "Mixed-signal computing for deep neural network inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 3–13, 2021.

[2] A. Pedram, S. Richardson, M. Horowitz, S. Galal, and S. Kvatinsky, "Dark memory and accelerator-rich system optimization in the dark silicon era," *IEEE Design Test*, vol. 34, no. 2, pp. 39–50, 2017.

[3] X. Liu, K. Katti, Y. He, P. Jacob, C. Richter, U. Schroeder, S. Kurinec, P. Chaudhari, and D. Jariwala, "Analog content-addressable memory from complementary fefets," *Device*, vol. 2, no. 2, p. 100218, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666998623003587

[4] I. O'Connor et al, "Library of optimized vnwfet-based logic cells, a eu deliverable." [Online]. Available: D4.01_FVLLMONTI_P3-ECL-INL-20230831

[5] E. T. Breyer, H. Mulaosmanovic, T. Mikolajick, and S. Slesazeck, "Reconfigurable nand/nor logic gates in 28 nm hkmg and 22 nm fd-soi fefet technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 28.5.1–28.5.4.

[6] Y. Xiao, Y. Xu, S. Deng, Z. Zhao, S. George, K. Ni, and V. Narayanan, "A compact ferroelectric 2t-(n+1)c cell to implement and-or logic in memory," in *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2023, pp. 1–6.

[7] S. De, F. Müller, N. Laleni, M. Lederer, Y. Raffel, S. Mojumder, A. Vardar, S. Abdulazhanov, T. Ali, S. Dünkel, S. Beyer, K. Seidel, and T. Kämpfe, "Demonstration of multiply-accumulate operation with 28 nm fefet crossbar array," *IEEE Electron Device Letters*, vol. 43, no. 12, pp. 2081–2084, 2022.

[8] Z. Jiang, Z. Zhao, S. Deng, Y. Xiao, Y. Xu, H. Mulaosmanovic, S. Duenkel, S. Beyer, S. Meninger, M. Mohamed, R. Joshi, X. Gong, S. Kurinec, V. Narayanan, and K. Ni, "On the feasibility of 1t ferroelectric fet memory array," *IEEE Transactions on Electron Devices*, vol. 69, no. 12, pp. 6722–6730, 2022.

[9] B. Jiang, Zurcher, Jones, Gillespie, and Lee, "Computationally efficient ferroelectric capacitor model for circuit simulation," in *1997 Symposium on VLSI Technology*, 1997, pp. 141–142.

# Compression ANS pour des réseaux de neurones

Olivier Romane, Olivier Muller, Adrien Prost-Boucle, Frédéric Pétrot
Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France

***Abstract*—Les réseaux de neurones artificiels posent des problèmes de performance lors de la phase d'inférence où le temps et l'énergie du calcul sont dominés par les accès aux mémoire externes. Nous proposons ici de réduire la quantité de données accédée en intégrant une étape de compression entropique lors des lectures mémoires. Nous étudions l'impact matériel des décompresseurs et montrons qu'il existe différents compromis architecturaux impactant le ratio de compression atteignable.**

***Index Terms*—compression de données, réseaux de neurones**

## I. Introduction

Les réseaux de neurones sont omniprésents dans le monde actuel, avec des réseaux comptant de plus en plus de paramètres notamment depuis l'avènement des LLMs (405 milliards de paramètres pour Llama3 [3] contre 25 millions pour Resnet50 [5]). Il devient donc nécessaire de construire du matériel adapté pour pouvoir réaliser l'entraînement et l'inférence de ces réseaux de façon efficace. La littérature a montré que les accès aux mémoires externes dominent le coût temporel et énergétique lors de l'exécution d'un réseau de neurones [4]. Ainsi, nous proposons d'exploiter la répartition inégale des poids des réseaux de neurones quantifiés pour appliquer une étape de compression entropique. Cela d'augmenter la quantité de données accédées à chaque lecture mémoire au prix d'une latence plus élevée due au calculs nécessaires pour la décompression.

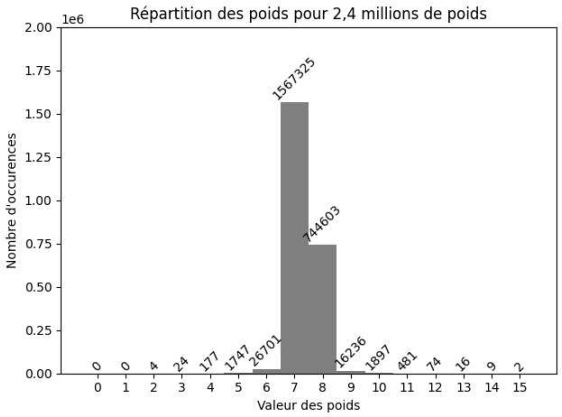



Fig. 1. Histogramme représentant la répartition des poids de la dernière couche de convolution d'un Resnet18 quantifié en 4 bits

*Institut National Polytechnique Grenoble Alpes

## II. Choix de l'algorithme de compression

Il est important de choisir une méthode de compression entropique qui admet à la fois une implantation matérielle simple tout en permettant de maximiser le ratio de compression des paramètres des réseaux. Une étude statistique de la répartition des poids de plusieurs réseaux de neurones denses quantifiés avec différentes méthodes de quantification montre que les poids sont fortement biaisés (Fig. 1).

L'algorithme de Huffman [6] admet une implantation matérielle raisonnable mais souffre d'un ratio de compression peu efficace pour les distributions de probabilité fortement déséquilibrées observées. De l'autre côté du spectre, le codage arithmétique [7] [9] fourni un ratio de compression efficace pour toute distribution de probabilité mais est coûteux pour une implantation matérielle. Un juste milieu pourrait être ANS [1] [2], et en particulier sa version tabulée tANS car il combine une complexité d'implantation raisonnable avec un ratio de compression efficace quelle que soit la distribution de probabilité de l'alphabet d'entrée.

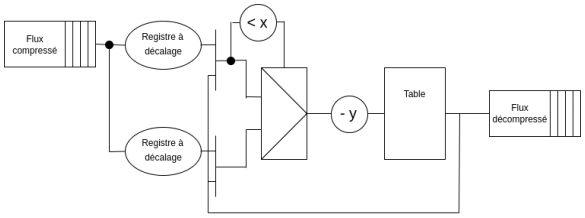



Fig. 2. Décompresseur tANS classique, $x$ et $y$ sont des constantes qui dépendent des paramètres du décompresseur

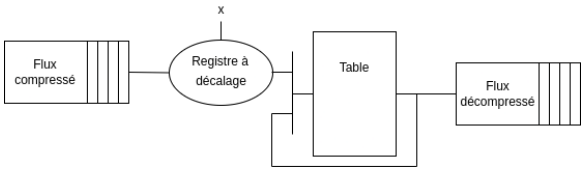



Fig. 3. Décompresseur tANS simplifié

## III. Implémentation et évaluation

L'algorithme de décompression de tANS est piloté par un automate d'états que l'on microcode dans la table d'états. Les données compressées sont contenues dans le flux d'entrée. Chaque ligne de la table d'états contient le symbole décompressé ainsi qu'une partie des bits du prochain état. Les bits du flux d'entrée contiennent l'autre partie des bits du prochain état.

L'algorithme de décompression de tANS est paramétré par le nombre d'états de l'automate et le nombre de symboles de l'alphabet compressé. Nous avons regroupé ces choix de paramètres en différentes classes, et on peut voir en Fig. 2, Fig. 3 que certaines classes de paramètres permettent de simplifier l'implantation du décompresseur. Cependant, cette simplification peut engendrer une perte sur le ratio de compression des décompresseurs comme montré dans la table I.

TABLE I
RATIO DE COMPRESSION (RC) POUR LA DERNIÈRE COUCHE DE CONVOLUTION DE RÉSEAUX QUANTIFIÉS RESNET18

| | Quantification 2 bits | Quantification 4 bits |
|---|---|---|
| RC Théorique | 2.15 | 3.77 |
| RC tANS classique | 2.13 | 3.71 |
| RC tANS simplifié | 2.13 | 3.43 |
| RC Huffman | 1.49 | 2.92 |

## IV. APPLICATION

Ce travail s'inscrit dans le cadre du projet Nnawaq [8], un accélérateur FPGA pour l'inférence de réseaux de neurones. L'architecture de Nnawaq est *dataflow*, c'est-à-dire que tous les poids du réseau implanté sont dans les mémoires internes du FPGA. Cela limite fortement la taille des réseaux qu'il est possible d'accélérer par cette méthode. Par conséquent, l'ajout de décompresseurs en sortie des mémoires internes du FPGA devrait permettre l'implantation de réseaux de neurones contenant plus de paramètres.

## V. CONCLUSION

Nous proposons de tirer partie de la répartition biaisée des poids des réseaux de neurones afin d'appliquer une passe de compression entropique à l'inférence, ce qui permet de réduire le nombre d'accès à des mémoires externes tout en offrant une implantation adaptée à l'architecture ciblée. Différentes implantations de tANS sont utilisées pour permettre d'atteindre différents compromis en terme d'impact architectural et de ratio de compression.

## REFERENCES


[1] Jarek Duda. Asymmetric numeral systems. *CoRR*, abs/0902.0271, 2009.

[2] Jarek Duda. Asymmetric numeral systems as close to capacity low state entropy coders. *CoRR*, abs/1311.2540, 2013.

[3] Aaron Grattafiori et al. The llama 3 herd of models, 2024.

[4] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[7] R. Pasco. Source coding algorithms for fast data compression (ph.d. thesis abstr.). *IEEE Transactions on Information Theory*, 23(4):548–548, 1977.

[8] Adrien Prost-Boucle, Alban Bourge, and Frédéric Pétrot. High-efficiency convolutional ternary neural networks with custom adder trees and weight compression. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11:1 – 24, 2018.

[9] J. J. Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

# An Analog MAC Cell in Standard CMOS for Neural Network Inference

Mohamed BOUCHAKOUR*†, Emmanuel BERGERET†, Gilles SICARD‡, Kamel ABDELOUAHAB§ and François BERRY*

*University Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France

†University Clermont Auvergne, CNRS, Laboratoire de Physique de Clermont Auvergne, F-63000 Clermont-Ferrand, France

‡CEA-LETI, F-38000 Grenoble, France

§Sma-RTy, F-63000 Clermont-Ferrand, France

Email: mohamed.bouchakour@uca.fr

***Abstract*—This article presents a fully analog Multiply-Accumulate (MAC) cell for neural network inference, implemented in standard CMOS technology. By exploiting the subthreshold exponential behavior of PMOS transistors modulated by the substrate, our design enables two-quadrant multiplication without the overhead of storing negative weights. The circuit thus achieves a significant reduction in energy consumption and circuit complexity while maintaining a high computational throughput. Simulations in a 65nm CMOS process show an energy consumption as low as 0.95 fJ per MAC operation and a total consumption of 0.92 pJ for inference on the MNIST dataset, with a classification rate of 91 %. Comparisons with the state of the art show notable improvements in energy efficiency.**

***Index Terms*—Analog computing, Multiply-Accumulate, Neural network inference, Low-power design, CMOS.**

## I. Introduction

Multiply–Accumulate (MAC) operations dominate the computational load in neural networks (NNs), often accounting for more than 99% of total operations. While digital implementations allow massive parallelism, they incur high energy consumption and latency [12].

Analog computing offers a promising alternative, enabling energy efficiency gains of several orders of magnitude [7]. However, analog designs face challenges such as process–voltage–temperature (PVT) variations, noise sensitivity, and the difficulty of storing analog weights. The intrinsic tolerance of neural networks to noise and approximation, combined with chip-specific training strategies, helps mitigate these issues.

Mixed-signal architectures combine analog computing with digital weight storage, using data reuse to amortize memory access costs despite frequent analog-to-digital conversions [10]. Eliminating these conversions through a fully analog approach promises additional efficiency gains. Although some designs exist [11], they remain much less common than mixed-signal architectures.

Several fully analog MAC cell implementations have been proposed: floating-gate current mirrors using stored charge and pulse-width modulation [9]; deep-well NMOS devices using simultaneous gate–substrate control [6]; dual-line schemes for positive/negative weights (doubling storage); near-memory systems using digital SRAM weights to drive in-cell DACs [8], [11]; and temporal-domain cells on SOI sharing back-gate control [8].

This project was funded by the Auvergne-Rhône-Alpes Region.

Furthermore, the widespread use of ReLU activation functions, producing non-negative activations [4], makes two-quadrant multiplication sufficient for most DNN layers, significantly simplifying hardware. In this work, we introduce a fully analog two-quadrant MAC cell based on substrate-modulated PMOS transistors in standard CMOS technology, removing the need for dedicated negative weight storage, reducing complexity, power, and area without loss of accuracy.

## II. Proposed Analog MAC Circuit

In standard CMOS, P-type transistors in an N-well allow independent substrate-bias control. In the subthreshold regime, the PMOS drain current is given by:

$$I_d = I_0 \cdot e^{-(1-k)V_{bs}/V_T} \cdot e^{-kV_{gs}/V_T} \cdot (1 - e^{-V_{ds}/V_T}) \quad (1)$$

Neglecting the Early effect and assuming saturation ($V_{ds} \geq 4V_T$), this simplifies to:

$$I_d = I_0 \cdot e^{-(1-k)V_{bs}/V_T} \cdot e^{-kV_{gs}/V_T} \quad (2)$$

This exponential dependence on $V_{bs}$ and $V_{gs}$ is exploited to perform multiplication in our circuit (Fig. 1), which comprises identical PMOS transistors P1, P2, and P3, and a metal–metal capacitor $C_{mem}$ charged via the bit line (BL).

For transistors P1 and P2, we have:

$$I_{in} = I_d(P1) = I_0 \cdot e^{-kV_{gsP1}/V_T} \quad (3)$$

$$I_{out1} = I_d(P2) = I_0 \cdot e^{-(1-k)V_{bs}/V_T} \cdot e^{-kV_{gsP2}/V_T} \quad (4)$$

Assuming $V_{gsP1} = V_{gsP2}$, we obtain:

$$I_{out1} = I_{in} \cdot e^{-(1-k)V_{bs}/V_T} \equiv I_{in} \cdot e^{-C \cdot V_{bs}} \quad (5)$$

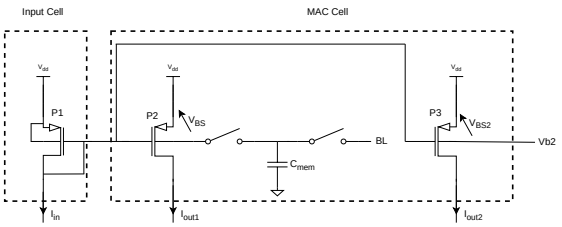



Fig. 1. Proposed MAC cell

We define the weight $w = e^{-C \cdot V_{bs}}$, with $w > 0$, so:

$$I_{out1} = I_{in} \cdot w \tag{6}$$

To support negative weights, a transformation $w = a \cdot w' + b$ is applied, yielding:

$$I_{out1} = \underbrace{I_{in} \cdot a \cdot w'}_{\text{Multiplication result}} + \underbrace{I_{in} \cdot b}_{\text{Output bias}} \tag{7}$$

Using a third transistor (P3) and taking the output as the difference between $I_{out1}$ and $I_{out2}$, we adjust $V_{bs2}$ such that $b = e^{-C \cdot V_{bs2}}$, canceling the bias:

$$I_{out} = I_{in} \cdot a \cdot w' \tag{8}$$

thus achieving two-quadrant multiplication. Placing multiple cells on the same output lines implements the sum of multiple products, enabling dot product computations fundamental to neural network inference.

## III. Tests and Main Results

The MAC cell was simulated in a 65nm CMOS process. A 784×10 MAC network used for MNIST classification achieved 91% accuracy.

The main contributors to power consumption in the system are the input cell and two computing transistors (P2 and P3). To maintain sub-threshold operation, the input current is capped at $1\mu A$, with an average of $0.5\mu A$ assumed under uniform input distribution. With a supply voltage of $0.6V$, the input cell consumes an average of $0.3\mu W$. However, this power is amortized over multiple MAC operations—in this case, distributed across 10 columns, effectively reducing its per-MAC contribution.

The two computing transistors also operate under similar assumptions, with an average input current of $0.5\mu A$ and a bias voltage of $0.6V$, leading to a combined power consumption of $0.6\mu W$. Since the energy per MAC depends on the computation time, a duration of 1.5ns results in an energy usage of $0.95fJ$ per MAC. This translates to an efficiency of $2105TOPS/W$, considering each MAC comprises two operations.

Table I compares our solution with the state of the art.

Custom training algorithms are needed to compensate practical nonidealities such as process variations and mismatch.

TABLE I
COMPARISON OF ANALOG MAC CELLS

| **Ref.** | [3] | [11] | [8] | [6] | [13] | [5] | **This work** |
|---|---|---|---|---|---|---|---|
| **Tech. (nm)** | 65 | 28 | 22 (FD-SOI) | 180 | 28 | 16 | **65** |
| **Type** | Time domain, A.C$^{(1)}$ | Time domain, A.C$^{(1)}$ | Time domain, A.C$^{(1)}$ | Substrate modulation$^{(3)}$ | CIM$^{(2)}$ | CIM$^{(2)}$ | **Substrate modulation**$^{(3)}$ |
| **Act. prec.** | 4/8 | A$^{(5)}$ | up to 6 | 4 | 1/2 | 1/8 | **A**$^{(5)}$ |
| **Weight prec.** | 4/8 | 5 | up to 6 | 4 | 1/2 | 1/8 | **5** |
| **Output type** | S$^{(4)}$ | S$^{(4)}$ | M$^{(5)}$ | M$^{(5)}$ | S$^{(4)}$ | S$^{(4)}$ | **M**$^{(5)}$ |
| **Eff. (TOPS/W)** | 102.2 | 437.9 | 2989* | 47.6 | 588 | 121/30 | **2105** |

$^{(1)}$ Charge accumulation. $^{(2)}$ Compute-in-memory. $^{(3)}$ Substrate modulation.
$^{(4)}$ Simulation results. $^{(5)}$ Measured results. $^{(6)}$ Analog.
* Calculated from published data.

## IV. Conclusion

We have designed a fully analog MAC cell in standard CMOS technology, achieving two-quadrant multiplication via subthreshold substrate modulation. Eliminating the need for negative weight storage significantly reduces complexity and power, achieving an energy efficiency of 2105 TOPS/W while maintaining 91% accuracy on MNIST. Future work includes full integration, error compensation, and chip-level validation.

## References


[1] Ali, M., et al., "IMAC: In-Memory Multi-Bit Multiplication and ACcumulation in 6T SRAM Array," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 8, pp. 2521–2531, Aug. 2020.

[2] A. G. Andreou *et al.*, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 205–213, Mar. 1991.

[3] Chen, Z., et al., "High-Throughput Dynamic Time Warping Accelerator for Time-Series Classification With Pipelined Mixed-Signal Time-Domain Computing," *IEEE J. Solid-State Circuits*, vol. 56, no. 2, pp. 624–635, Feb. 2021.

[4] Dubey, S.R., Singh, S.K., Chaudhuri, B.B.: Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing.* 2022; **503**:92–108.

[5] Jia, H., et al., "Scalable and Programmable Neural Network Inference Accelerator Based on In-Memory Computing," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 198–211, Jan. 2022.

[6] Kenarangi, F., et al., "A Single-MOSFET Analog High Resolution-Targeted (SMART) Multiplier for Machine Learning Classification," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 11, no. 4, pp. 816–828, Dec. 2021.

[7] B. Murmann, "Mixed-Signal Computing for Deep Neural Network Inference," *IEEE Trans. VLSI Syst.*, vol. 29, no. 1, pp. 3–13, Jan. 2021.

[8] Nägele, R., et al.: Analog Multiply-Accumulate Cell With Multi-Bit Resolution for All-Analog AI Inference Accelerators. *IEEE Transactions on Circuits and Systems I: Regular Papers.* 2023 Sep; **70**(9):3509–21.

[9] Paliy, M., Strangio, S., Ruiu, P., Rizzo, T., Iannaccone, G.: Analog Vector-Matrix Multiplier Based on Programmable Current Mirrors for Neural Network Integrated Circuits. *IEEE Access.* 2020; **8**:203525–37.

[10] Sebastian, A., et al., "Memory devices and applications for in-memory computing," *Nat. Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020.

[11] Seo, J.O., et al.: A 44.2-TOPS/W CNN Processor With Variation-Tolerant Analog Datapath and Variation Compensating Circuit. *IEEE Journal of Solid-State Circuits.* 2024 May; **59**(5):1603–11.

[12] V. Sze, et al., "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[13] Zhang, B., et al., "PIMCA: A Programmable In-Memory Computing Accelerator for Energy-Efficient DNN Inference," *IEEE J. Solid-State Circuits*, vol. 58, no. 5, pp. 1436–1449, May 2023.

# Estimation temps réel de pagayage par capteurs de force

Souébou Bouro[1], Mickaël Le Gentil[1], Antoine Courtay[1], Guillaume Nicolas[2], Nicolas Bideau[2], Olivier Berder[1]
[1]University of Rennes - IRISA, firstname.lastname@irisa.fr
[2]University of Rennes 2 - M2S, firstname.lastname@univ-rennes2.fr

## I. Introduction

L'analyse de la performance en kayak repose sur la mesure de paramètres dynamiques tels que la force appliquée sur la pagaie et la cadence de pagayage, définie comme le nombre de coups par minute (SPM). Ces indicateurs sont essentiels pour évaluer l'efficacité du geste et adapter l'entraînement. Les ergomètres offrent un environnement de mesure contrôlé [1], [2], mais ne reproduisent que partiellement les conditions réelles de navigation. L'analyse vidéo 3D [3] permet de suivre le geste technique avec précision, mais reste peu adaptée aux conditions extérieures.

La comparaison de la cadence à d'autres métriques, comme la puissance ou les données d'ergomètre [4], ou un métronome sonore [5], soulignent l'importance de cette métrique sans pour autant proposer de solution adaptée à une estimation en temps réel. Par ailleurs, une approche classique d'estimation de la cadence basée sur la détection de pics de force serait sensible aux variations interindividuelles (technique, niveau, environnement).

Dans ce contexte, nous proposons une approche temps réel reposant uniquement sur les signaux de force mesurés sur la pagaie. L'algorithme se repose sur l'identification de la fréquence fondamentale du signal de force à travers une fonction d'autocorrélation (ACF) adaptée aux spécificités du pagayage, intégrée dans un système embarqué léger et économe en énergie.

## II. Méthodologie d'estimation de la cadence

Plusieurs approches sont envisageables pour l'estimation de cette fréquence fondamentale, selon le domaine d'analyse considéré. Les méthodes fréquentielles (FFT, cepstre, spectre harmonique) offrent une bonne précision sur les signaux stationnaires, mais leur coût computationnel élevé et leur sensibilité au bruit les rendent peu adaptées aux systèmes embarqués temps réel.

En revanche, les méthodes temporelles sont plus légères et directement exploitables sur microcontrôleurs. L'autocorrélation (ACF), l'Average Magnitude Difference Function (AMDF) et les variantes simplifiées [6], [7] offrent un bon compromis entre précision, robustesse et efficacité. L'ACF est adaptée ici à la dynamique spécifique du pagayage, avec un traitement par fenêtres glissantes, un filtrage léger, et une détection optimisée du pic principal corrélé à la période fondamentale.

### A. Instrumentation de la pagaie

Le système de mesure développé repose sur deux modules embarqués (Fig. 1-b) fixés de part et d'autre de la pagaie, chacun intégrant une jauge de contrainte collée à 65 cm du centre du manche (Fig 1-a). La déformation mesurée est convertie en tension à l'aide d'un pont de Wheatstone et numérisée par un convertisseur analogique-numérique (ADC) 24 bits à 80 Hz, avec suréchantillonnage logiciel à 100 Hz pour transmission Bluetooth. Une référence externe de 122 mV est utilisée pour adapter la pleine échelle aux faibles variations de résistance des jauges (±2 Ω).

Les modules sont compacts (52x28x15 mm, 15 g ($< 180g$ [4])), synchronisés grâce à un horodatage partagé, et n'affectent pas l'équilibre ou la maniabilité de la pagaie. Les données sont transmises en temps réel vers une tablette en Bluetooth pour traitement logiciel.

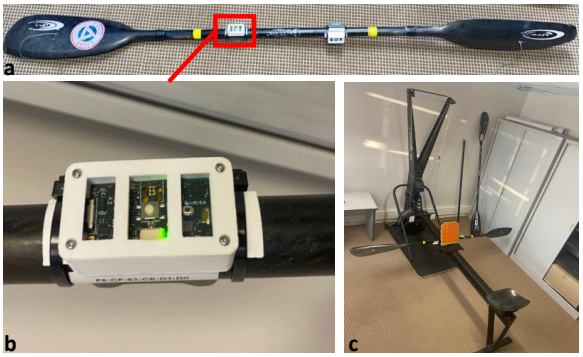



Fig. 1: Instrumentation de la pagaie : a) Pagaie instrumentée, b) Nœud capteur, c) Ergomètre.

### B. Protocole d'évaluation

L'algorithme est conçu pour couvrir une plage de cadence allant de 30 à 180 SPM. Une première validation est réalisée à l'aide de signaux sinusoïdaux simulant différentes cadences (30 à 180 SPM par pas de 1 SPM), sur des fenêtres glissantes de 4 secondes avec recouvrement de 3,5 secondes.

Ensuite, des tests sont réalisés sur ergomètre (Fig 1-c) avec des athlètes à différentes intensités : Faible (LSR, 30-67 SPM), Modérée (MSR, 68-105 SPM), Élevée (HSR, 106-143 SPM) et Maximale (XSR, 144-180 SPM). Les données de notre système (mise à jour chaque 0.5 s) sont comparées aux données de cadence de l'ergomètre (mise à jour 0.4–0.7 s), après interpolation linéaire.

Les performances sont évaluées à l'aide du coefficient de corrélation de Pearson ($r$), de l'erreur quadratique moyenne (RMSE).

## III. Validation expérimentale sur ergomètre

La Figure 2 compare l'ACF classique à notre approche optimisée avec bornes dynamiques. Cette adaptation permet de limiter le calcul aux décalages pertinents tout en préservant la précision. Ce traitement est robuste, peu sensible aux variations individuelles et compatible avec une exécution embarquée. L'ajustement dynamique des bornes $K_{min}$, ($K_{ref}$, ($K_{max}$ en fonction du taux d'échantillonnage effectif permet d'optimiser à la fois la précision et l'efficacité computationnelle. L'approche est ainsi bien adaptée aux contraintes des systèmes embarqués sportifs, avec des performances stables sur une large plage de cadences.

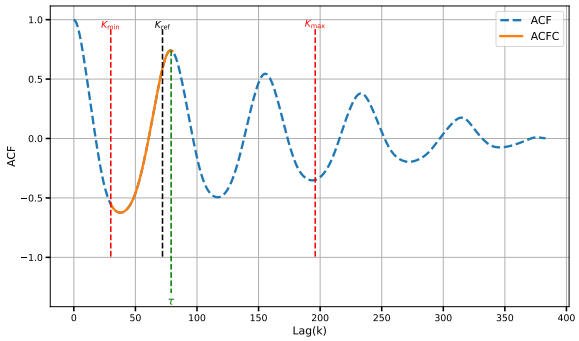



Fig. 2: Comparaison entre l'ACF classique et notre version avec ajustements dynamiques (ACFC).

La table I présente la moyenne des résultats des tests sur ergomètre. Des écarts initiaux de cadence sont observés lors des premiers coups (1–5), probablement liés à l'accélération du volant d'inertie, connue pour fausser les mesures de l'ergomètre [8]. Ces coups ont été exclus de l'analyse. En effet, On observe une forte corrélation ($r$ entre 0,89 et 0,97) et des RMSE compris entre 3,26 et 7,40 SPM selon les plages de cadence. Les écarts les plus importants apparaissent naturellement aux hautes cadences, à cause d'une baisse de résolution relative.

TABLE I: Moyennes des résultats de performance pour un athlète professionnel

| Condition | Corrélation $r$ | RMSE (SPM) |
|---|---|---|
| LSR | 0,93 | 3,65 |
| MSR | 0,97 | 3,26 |
| HSR | 0,89 | 7,36 |
| XSR | 0,92 | 7,40 |

La Figure 3 illustre la dynamique de la force enregistrée et l'évolution des cadences estimées comparées à celles de l'ergomètre. Globalement, le système fournit des estimations précises, stables et compatibles avec les exigences d'analyse de performance. La méthode se montre robuste sans nécessiter d'ajustement individuel ni connaissance a priori du niveau technique.

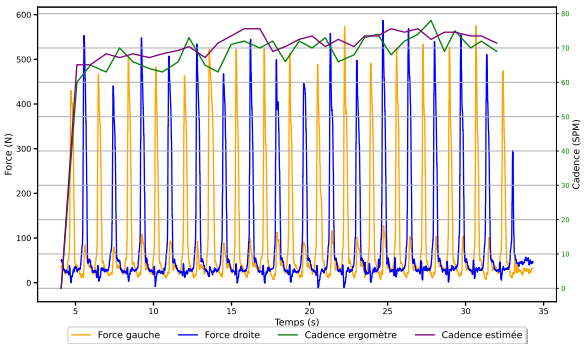



Fig. 3: Force mesurée par la pagaie instrumentée et comparaison des cadences estimées avec celles de l'ergomètre.

## IV. Conclusion

Nous avons présenté un système embarqué permettant d'estimer en temps réel la cadence de pagayage à partir de capteurs de force positionnés sur la pagaie. L'algorithme repose sur une fonction d'autocorrélation (ACF) adaptée, combinée à une chaîne de prétraitement légère et optimisée pour un traitement sur microcontrôleur.

La méthode est robuste, et ne dépend d'aucun paramètre utilisateur. Les tests sur signaux sinusoïdaux ont permis de caractériser la résolution en fonction de la cadence. Les expérimentations sur ergomètre, menées à différentes cadences, ont confirmé la fiabilité des estimations, avec une forte corrélation aux données de référence.

Ce système ouvre des perspectives pour l'analyse de la performance en kayak, en particulier en conditions réelles sur l'eau. Les travaux futurs viseront à intégrer d'autres capteurs (orientation, vitesse, force sur le cale-pied).

## References


[1] M. Begon, F. Colloud, and P. Lacouture, "Measurement of contact forces on a kayak ergometer with a sliding footrest–seat complex," *Sports engineering*, vol. 11, pp. 67–73, 2009.

[2] P. Bonito, M. Sousa, F. J. Ferreira, J. F. Justo, and B. Gomes, "Magnitude and shape of the forces applied on the foot rest and paddle by elite kayakers," *Sensors*, vol. 22, no. 4, p. 1612, 2022.

[3] E. Limonta, R. Squadrone, R. Rodano, A. Marzegan, A. Veicsteinas, G. Merati, and M. Sacchi, "Tridimensional kinematic analysis on a kayaking simulator: key factors to successful performance," *Sport Sciences for Health*, vol. 6, pp. 27–34, 2010.

[4] D. Sturm, K. Yousaf, and M. Eriksson, "A wireless, unobtrusive kayak sensor network enabling feedback solutions," in *International Conference on Body Sensor Networks*, 2010, pp. 159–163.

[5] B. Gomes, N. V. Ramos, F. A. Conceição, R. H. Sanders, M. A. Vaz, and J. P. Vilas-Boas, "Paddling force profiles at different stroke rates in elite sprint kayaking," *Journal of Applied Biomechanics*, vol. 31, no. 4, pp. 258–263, 2015.

[6] L. Rabiner, M. Cheng, A. Rosenberg, and C. McGonegal, "A comparative performance study of several pitch detection algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 5, pp. 399–418, 1976.

[7] Y.-M. Zeng, Z.-Y. Wu, H.-B. Liu, and L. Zhou, "Modified amdf pitch detection algorithm," in *International Conference on Machine Learning and Cybernetics*, vol. 1, 2003, pp. 470–473.

[8] G. Treff, L. Mentz, B. Mayer, K. Winkert, T. Engleder, and J. M. Steinacker, "Initial evaluation of the concept-2 rowing ergometer's accuracy using a motorized test rig," *Frontiers in Sports and Active Living*, vol. 3, p. 801617, 2022.

# Plug, Play, Betray: Edge TPU Under Man-in-the-Middle Attack

S. O. Niang, J. Lorandel and C. Moy
*Univ Rennes, CNRS, IETR UMR 6164, F-35000, Rennes, France*
seydina-oumar.niang@univ-rennes.fr

***Abstract*—AI-based hardware accelerators such as the Coral USB Accelerator enable the use of AI on the edge nowadays. They offer many advantages over cloud-based solutions. However, the lack of encryption and integrity checks in USB communication raises security concerns, as these accelerators can be used in critical applications. In this paper, we present a man-in-the-middle (MITM) attack targeting the USB interface of the Coral USB Accelerator. Our methodology is based on the Raspberry Pi 5 and USB Proxy [1], allowing real-time interception and manipulation, and thus enabling us to successfully demonstrate the exfiltration of sensitive data (model parameters, input data) and the corruption of inference results.**

***Index Terms*—Edge TPU, Man-in-the-middle attack, USB security, Hardware vulnerability, Data exfiltration**

## I. Introduction

Nowadays, AI is used in many applications (over 400 specific use cases, according to McKinsey [2]), including critical fields like healthcare, industry, and finance. Most of these applications rely on cloud servers to perform the heavy computations required by deep neural networks (DNNs). Now, Edge AI-based hardware accelerators (such as Google Edge TPUs [3]edge tpus) make it possible to bring the power of AI to embedded systems without relying on cloud servers. Deploying DNNs on edge devices has many advantages [4], including low latency, low power consumption, and enhanced privacy. However, to interface the accelerator with the host system, the communication interfaces often rely on PCIe and USB. It has been shown that the neural network models they run through them were vulnerable to attacks. In particular, quite a few attacks targeting edge-deployed DNNs have been published. Isakov et al. [4] proposed a survey that classifies them based on the attacker's access level and agenda. On the other hand, Wu et al. [5] were able to intercept and manipulate image data transmitted between a USB camera and a detection system. Considering all these works and the fact that AI accelerators are used in critical areas, it is essential to assess their security. Even high-performance buses such as PCIe can also be the target of hardware attack, allowing direct access to exchanged data. In [6], a man-in-the-middle (MITM) attack was demonstrated on the PCIe bus of an iPhone, which connects the SoC to its NAND flash memory, allowing them to bypass the lock screen's password attempt limit. In this paper, a man-in-the-middle (MITM) attack targeting the USB interface of Google's Coral USB Accelerator is demonstrated. The proposed approach differs from previous works [5] [7], in which the MITM attack is performed between the host and the input device. In this work, the attack is performed between the host and the Coral USB Accelerator. While previous attacks were only targeting input data, the proposed approach allows an attacker to access all data exchanged between the host and the accelerator, including input, model parameters, and inference results, making possible a large panel of attacks.

This study is partially funded by the ANR within the framework of the PIA EUR CyberSchool project (ANR-18-EURE-0004).


Fig. 1. USB MitM Attacks on the Coral USB Accelerator

## II. MITM Attack on USB communication bus

### A. Threat model

In our threat model, the following assumptions are made: the Coral USB Accelerator is used in an image classification application; the attacker has physical access to the USB cable connecting the host and the accelerator and can insert a malicious device between the Coral USB Accelerator and the host; neither the host nor the accelerator is compromised or damaged; and there is no encryption or integrity check on the USB traffic.

### B. Methodology

Most AI computation is all about tensor calculations, and the Edge-Tensor Processing Unit (TPU), the main component of the Coral USB Accelerator, is specially designed to accelerate these operations. It is a custom ASIC that employs a systolic array of processing elements (PEs) organized into a 2D grid. The Coral USB Accelerator is designed to be used as a coprocessor. That's why it is always used in conjunction with a host system through a USB interface. Our approach (Figure 1) consists of using a Raspberry Pi 5 with USB Proxy, an open-source software project that supports USB proxying with options to perform MITM attacks, to gain full control over the USB traffic between the host and the Coral USB Accelerator. In the literature, hardware platforms used to perform MITM attacks, even outside the USB context, are mainly based on FPGAs [7] [6]. Cynthion [8] is one of them and is specifically designed for USB analysis and can be used to perform MITM

attacks when combined with a host. In this work, USB Proxy [1] was reused to ensure the MITM on the USB link, allowing us to focus on the reverse engineering of the edge-TPU runtime and the implementation of the attack scenarios.

### C. Reverse engineering the USB traffic

A preliminary step was to passively capture the USB traffic and analyze it in order to identify the different types of transactions exchanged between the host and the accelerator. To this purpose, Wireshark was used to monitor all the traffic in real time. After being able to see what is being sent and received in a decoded format, we moved forward with the understanding of the Edge TPU Runtime (libedgetpu), the one in charge of all the data transfers. By Recompiling a new binary of the Edge TPU Runtime with debug symbols enabled and using gdb to step through the code during the inference process was done to better understand the sequence. In addition, the work of George Hotz on the reverse engineering of Coral-AI Edge-TPU was inspiring. In fact, he has tried to make his deep learning framework called Tynigrad, compatible with the Coral USB Accelerator in order to bypass completely the Coral USB Accelerator's software stack. From his work, we successfully enabled a multi-level logging system which was disabled by default. This was a critical step to understand what the Coral-AI runtime, called libedgetpu, was doing during the inference process. During the reverse engineering step, different types of transactions exchanged between the host and the accelerator based on the size of the allocated memory for each transaction were identified. Then, we were able to know how the host tells the accelerator that the data being sent corresponds to the input tensor, model parameters, or inference results. For instance, before each transaction, the host sends a bulk transfer with a payload of 8 bytes that contains the type of the next transaction to be sent and the size of its payload. This only applies to data sent from the host to the device. By knowing all these details, the attack scenarios can be refined.

### D. Attack scenarios

Being able to interface the USB link and to gain full control of the traffic, make it possible to develop a large panel of attacks. In this paper, two attack scenarios were considered: the first one consists in exfiltrating the model parameters and the input data. The second one aims at corrupting the inference results. For the first attack scenario, the USB Proxy's base code is modified so that both the input data and the model parameters are copied before being forwarded to the Coral USB Accelerator. The challenge is to know exactly when to stop copying and consider the copy complete, since the endpoint's maximum packet size is 512 bytes, while the input and model parameters can be much larger. The solution lies in the fact that the transaction size is known in advance, thanks to the 8-byte packet sent by the host before each transfer. As for the second attack scenario, the output was targeted directly. To do this, we simply intercept and tamper with the bulk-type transactions sent by the Accelerator back to the host through endpoint 0x81.

## III. Experimental Results

All two scenarios were successfully validated in our experiments. For the first scenario, model parameters and input data were exfiltrated without detection. The input image (parrot.jpg), for example, was easily regenerated from the raw byte stream copied. In the second scenario, the baseline correctly classified `parrot.jpg` as "Ara macao" (75.78% confidence, Table I). However, after modifying just the first byte of the output, the classification wrongly shifted to "Aeronautes saxatalis" with high confidence (99.61%, Table II), clearly highlighting the wrong classification with high confidence.

TABLE I
BASELINE INFERENCE RESULT FOR SAMPLE INPUT (`PARROT.JPG`).

| Input Image | Attack | Baseline Output (Class) | Baseline Output (Confidence) |
|---|---|---|---|
| parrot.jpg | Baseline (No Attack) | Ara macao | 0.75781 |

TABLE II
INFERENCE RESULT FOR `PARROT.JPG` AFTER TAMPERING FIRST BYTE OF OUTPUT TENSOR.

| Input Image | Attack | Tampered Output (Class) | Tampered Output (Confidence) |
|---|---|---|---|
| parrot.jpg | Set 0xFF the first byte of the tensor | Aeronautes saxatalis | 0.99609 |

## IV. Conclusion

Our research demonstrates critical vulnerabilities in USB communications of Edge AI accelerators. Using accessible tools, we revealed significant risks of data leakage and inference corruption, highlighting the urgent need for enhanced security measures in AI deployments.

## References


[1] Masataka Sawahara Aristo Chen, Andrey Konovalov. usb-proxy: A usb proxy based on raw-gadget and libusb. https://github.com/AristoChen/usb-proxy, 2024. Commit `c6454ce`; accessed Apr 18, 2025.

[2] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. Notes from the ai frontier: Insights from hundreds of use cases. *McKinsey Global Institute*, 2(267):1–31, 2018.

[3] Coral. Coral products. https://coral.ai/products/. Accessed: March 31, 2025.

[4] Mihailo Isakov, Vijay Gadepally, Karen M Gettings, and Michel A Kinsy. Survey of attacks and defenses on edge-deployed neural networks. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2019.

[5] Han Wu, Sareh Rowlands, and Johan Wahlström. A human-in-the-middle attack against object detection systems. *IEEE Transactions on Artificial Intelligence*, 2024.

[6] Mohamed Amine Khelif, Jordane Lorandel, Olivier Romain, Matthieu Regnery, Denis Baheux, and Guillaume Barbu. Toward a hardware man-in-the-middle attack on pcie bus. *Microprocessors and Microsystems*, 77:103198, 2020.

[7] Wenye Liu, Weiyang He, Bowen Hu, and Chip-Hong Chang. A practical man-in-the-middle attack on deep learning edge device by sparse light strip injection into camera data lane. In *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, pages 1–6. IEEE, 2022.

[8] Great Scott Gadgets. Cynthion. https://greatscottgadgets.com/cynthion/. Accessed: April 1, 2025.

# Implémentation de l'algorithme AKAZE en SYCL pour multicœur

Sonia Haddouche*, Erwan Fabiani†, Loïc Lagadec‡, Franck Danober§, Robin Lembach§, Christophe Guillet§
*Thales LAS France SAS Élancourt, ENSTA, Lab-STICC, CNRS, UMR 6285, Brest, France, sonia.haddouche@ensta.fr
†Univ Brest, Lab-STICC, CNRS, UMR 6285, Brest, France, erwan.fabiani@univ-brest.fr
‡ENSTA, Lab-STICC, CNRS, UMR 6285, Brest, France, loïc.lagadec@ensta.fr
§Thales LAS France SAS, Élancourt, France, {franck.danober, robin.lembach, christophe.guillet}@fr.thalesgroup.com

***Résumé*—Cette communication présente une étude de cas de programmation parallèle de l'algorithme AKAZE en SYCL sur multicœur, réalisée dans le cadre d'un projet initié au sein du laboratoire commun Lateral Lab-STICC/Thales-LAS. Il explique la méthodologie utilisée pour migrer un programme OpenMP vers SYCL.**

***Index Terms*—AKAZE, traitement d'image, programmation hétérogène, SYCL, multicœur.**

## I. Introduction

Disposer de capacités de calcul haute performance est aujourd'hui critique pour faire face à l'accroissement, tant en volume qu'en complexité, des données à traiter. Les plateformes hétérogènes qui permettent la mise en œuvre de calculs répartis sur des architectures matérielles différentes nécessitent un environnement de programmation facilitant la maitrise des différents modèles de calcul sous-jacents et leurs interactions.

Le standard SYCL [1] (du consortium Khronos) répond à cet enjeu. Basé sur du C++ mono-source, il met en œuvre la portabilité et l'hétérogénéité, via un modèle normalisé de tâches de calcul (kernels) communiquant par des tampons mémoires, dont le support d'exécution est paramétrable. Par conséquent, SYCL offre un environnement adéquat pour la définition de programmes hétérogènes qui exploitent conjointement plusieurs cibles architecturales (notamment les CPU multicœurs, les GPU et les FPGA).

Dans ce contexte, nous commençons un projet au sein du laboratoire commun Lateral Lab-STICC/Thales-LAS, qui vise à évaluer les apports de la programmation SYCL pour les traitements embarqués spécifiques au domaine métier considéré (traitement du signal, traitement d'image, classification), et à établir des méthodes de migration de code legacy et d'aide à la conception pour SYCL.

Comme premier benchmark, nous avons choisi l'algorithme AKAZE en raison de sa représentativité et de son utilisation dans des produits industriels.

Les deux principales implémentations actuelles de la spécification SYCL sont DPC++ [2] (qui fait partie de l'environnement de programmation OneApi [3]), et AdaptiveCPP (anciennement hypSYCL) [4]. Pour cette étude de cas, nous utilisons DPC++, avec comme support d'exécution un processeur multicœurs.

## II. Présentation de l'algorithme AKAZE

AKAZE [5] est un algorithme de traitement d'image pour l'extraction de caractéristiques (pixels clefs), qui permet un suivi 2D d'un objet dans un flot vidéo avec une bonne tolérance aux transformations géométriques. C'est une amélioration, en termes de temps de calcul, de l'algorithme KAZE [6]. La différence étant qu'AKAZE utilise une diffusion FED (Fast Explicit Diffusion) alors que KAZE utilise une diffusion non linéaire traditionnelle (AOS), qui conserve mieux les structures de l'image, mais nécessite une quantité de calculs bien plus importante.

L'algorithme AKAZE se déroule en plusieurs étapes principales :

1) Construction de l'espace à échelle non linéaire : une pyramide d'images est générée en appliquant la diffusion rapide (FED) pour obtenir différentes résolutions de l'image d'entrée.
2) Extraction des caractéristiques : des points d'intérêt sont détectés à différentes échelles en analysant les variations locales de l'intensité de l'image.
3) Description des caractéristiques : chaque point d'intérêt est caractérisé par un descripteur binaire robuste, permettant de résumer son apparence locale de manière compacte et efficace.
4) Appariement des caractéristiques : les descripteurs extraits sont comparés entre deux images afin d'identifier les correspondances potentielles entre les points d'intérêt.

Le résultat de l'algorithme, illustrant l'appariement des points caractéristiques entre deux images, est présenté dans la figure 1.

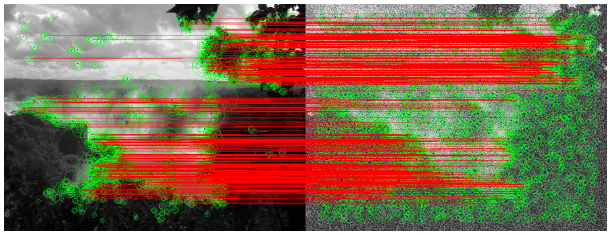


FIGURE 1. Appariement des points caractéristiques entre deux images.

## III. Processus de migration de AKAZE en SYCL

Nous avons choisi l'implémentation AKAZE de C. Sweeney [7] comme programme de référence. Sa particularité, par rapport à la version C++ originale [8], est qu'elle n'utilise pas la bibliothèque OpenCV, ce qui répond à nos contraintes industrielles. Cette version utilise la bibliothèque d'algèbre linéaire Eigen et exploite la parallélisation des boucles via des directives OpenMP.

Le processus de migration en SYCL DPC++ est constitué des étapes suivantes :

1) Adapter le code C++ aux standards modernes, remplacer les anciennes pratiques comme memcpy par des méthodes sûres comme copy, adaptées aux objets complexes.
2) Identifier et isoler les principaux noyaux de calcul de la version initiale, enregistrer les entrées/sorties de chacun d'entre eux, afin d'avoir une base de comparaison avec la version SYCL, en mettant en place des tests unitaires. La figure 2 illustre la structuration et l'organisation avec les noyaux identifiés.
3) Résoudre les problèmes de compatibilité entre la modélisation de la bibliothèque Eigen et l'implémentation de SYCL, notamment en utilisant la bibliothèque BLAS de oneMKL [9] de oneAPI pour calculer le produit matriciel à la place de celui d'Eigen, par exemple.
4) Modifier progressivement chacun des noyaux identifiés pour chaque étape clé de l'algorithme, les valider via les tests unitaires, puis les intégrer dans le programme global.
5) Comparer les temps de traitement des noyaux de calcul OpenMP et SYCL individuellement et globalement, en faisant varier le nombre de threads, afin d'identifier en amont toute anomalie dans l'exploitation du parallélisme.

Des résultats préliminaires sur plusieurs noyaux montrent des temps de calcul comparables à la version OpenMP. La répartition des work-items en SYCL offre plus de flexibilité que OpenMP, mais peut avoir un impact significatif sur les performances selon la structuration choisie. Ces deux aspects sont détaillés dans le poster.

## IV. Conclusion et perspectives

SYCL est un environnement de mise en œuvre de traitements accélérés parallèles multi-plateforme. Généraliser son usage dans des systèmes industriels amènerait un avantage de portabilité, mais cela nécessite de s'assurer de ses performances par rapport à du code parallèle existant, comme en OpenMP, et d'établir une méthodologie de migration automatisable. L'étude de cas présentée, l'algorithme AKAZE mis en œuvre sur multicœur avec DPC++, contribue à cette perspective.

La suite des travaux portera sur l'évaluation des performances avec l'implémentation AdaptiveCPP, puis l'étude de la portabilité du code pour une cible GPU avec une analyse des modifications spécifiques pour ce support. Cette étude de cas contribuera à établir des patrons de conception adaptés aux différentes architectures, dans le cadre de notre projet d'environnement d'aide à la conception SYCL.

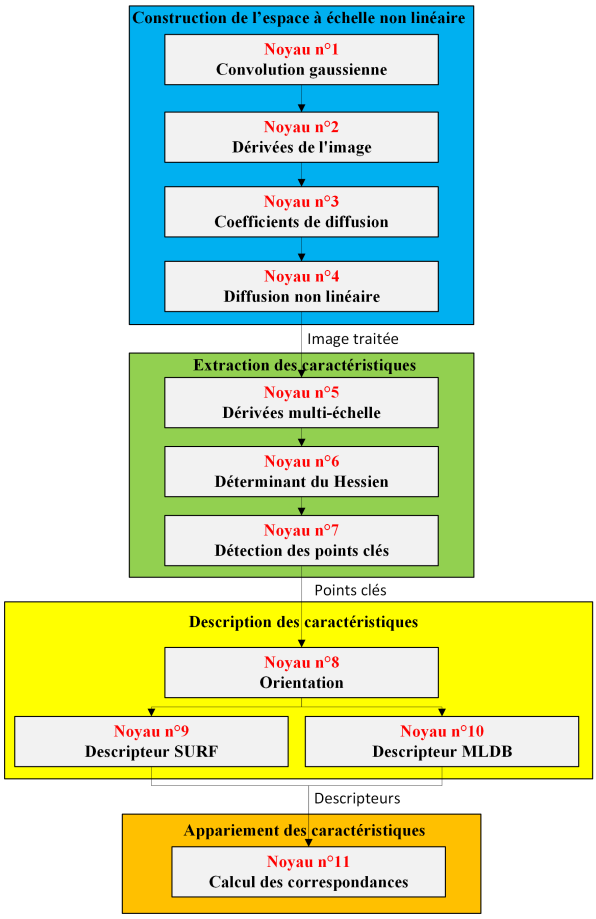



FIGURE 2. Organisation du code avec les noyaux identifiés.

## Références


[1] Khronos Group, "SYCL 2020 specification," 2020, https://registry.khronos.org/SYCL/specs/sycl-2020/pdf/sycl-2020.pdf

[2] J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, and X. Tian, *Data Parallel C++ : Mastering DPC++ for Programming of Heterogeneous Systems Using C++ and SYCL*, Springer Nature, 2021.

[3] oneAPI Programming Model. [En ligne]. https://www.oneapi.io/.

[4] A. Alpay and V. Heuveline, "AdaptiveCpp Stdpar : C++ Standard Parallelism Integrated Into a SYCL Compiler," in *Proceedings of the 12th International Workshop on OpenCL and SYCL (IWOCL '24)*, 2024.

[5] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *British Machine Vision Conference (BMVC)*, Bristol, UK, Sep. 2013.

[6] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *European Conference on Computer Vision (ECCV)*, Florence, Italy, Oct. 2012, pp. 624–638.

[7] C. Sweeney. *akaze-eigen*. Dépôt GitHub, https://github.com/sweeneychris/akaze-eigen, 2014.

[8] P. F. Alcantarilla. *akaze*. Dépôt GitHub, https://github.com/pablofdezalc/akaze, 2013.

[9] oneAPI Math Kernel Library. [En ligne]. https://oneapi-spec.uxlfoundation.org/specifications/oneapi/v1.3-rev-1/elements/onemkl/source/.

# FPGA-Based Framework for Memory Exploration in Heterogeneous AI Systems

Felipe PAIVA ALENCAR, Pascal BENOIT, and David NOVO
LIRMM, Univ. Montpellier, CNRS, Montpellier, France

***Abstract*—We propose a methodology to evaluate heterogeneous AI systems in the presence of emerging on-chip memory technologies with limited data-retention times. Our approach couples an in-house RISC-V microcontroller platform (ADAM) and a configurable systolic accelerator (Gemmini-SV) on an FPGA, integrated with the IREE compiler stack. By modeling emerging memories (e.g., capacitorless two-transistor cells) with synthesizable RTL modules and deploying them on the FPGA for cycle-accurate emulation, we aim to highlight performance, energy and trade-offs, as well as scheduling constraints introduced by limited retention times. This paper outlines the motivation, core methodology, and future directions of our FPGA-based exploration framework.**

***Index Terms*—Heterogeneous computing, FPGA, emerging memory, AI acceleration, scheduling**

## I. Introduction

The growth in machine learning workload size has motivated the development of specialized compute accelerators such as GPUs and TPU-like accelerators. This resulted in heterogeneous systems that integrate traditional CPUs with accelerators, boosting performance for these specialized workloads. However, coordinating computation, data movement, and memory use efficiently across these diverse resources is challenging. Adding to these coordination challenges, on-chip memory technologies that promise higher density (e.g., capacitorless two-transistor cells) at the cost of limited retention times impose strict constraints on data reuse and scheduling. We propose an FPGA-based framework to study these challenges through prototype hardware, compiler-level scheduling experiments, and emulated memory modules. This framework helps evaluate and optimize scheduling policies under limited-retention conditions.

## II. Related Works

High-level simulation frameworks enable rapid exploration of scheduling and data movement, but lack cycle-level precision. Frameworks such as Stream [1] support experiments on single and multiple cores with various data movement patterns.

SoC generator frameworks provide comprehensive environments for designing and evaluating heterogeneous systems. Projects like Chipyard [2] use Chisel, a non-industry-standard HDL, which raises the barrier for hardware modification to meet specific experimental needs.

Accelerator frameworks offer full-stack environments for prototyping systolic-array-based AI computations. The Gemmini accelerator [3] is a parametric, synthesizable systolic-array generator for RISC-V, enabling efficient hardware prototyping for AI kernels.

Compiler infrastructures bridge high-level ML models and hardware backends. IREE [4] is an MLIR-based compiler and runtime that targets CPUs and GPUs and can be extended to support custom accelerators.

Lu et al. observe that if retention times exceed the lifespan of activation data in typical AI workloads, explicit refresh operations become unnecessary, since data are overwritten before any decay occurs [5]. These studies often focus on device-level properties or high-level simulations rather than end-to-end system integration.

In contrast, our work deploys hardware prototypes—including a parametric RISC-V microcontroller, our Gemmini-SV accelerator (a SystemVerilog rewrite of Gemmini), and synthesizable RTL models of emerging memories—on a single FPGA platform, and couples them with host-executed, state-of-the-art compiler-driven scheduling workflows. This end-to-end setup enables comprehensive evaluation of scheduling strategies and hardware trade-offs under limited-retention conditions.

## III. Methodology

Our methodology comprises four key components: FPGA-based prototyping, in-house architecture modules, an MLIR-driven software stack, and RTL models of emerging memories. The following subsections describe each component in detail and show how they integrate into an end-to-end framework for evaluating scheduling strategies.

### A. FPGA-based Prototyping

We use FPGAs to achieve fast, cycle-accurate results without the lengthy runtimes of full RTL simulation. This approach delivers the speed of high-level simulators such as Stream without sacrificing accuracy. Our synthesizable, parametric architecture supports rapid reconfiguration of cores, accelerators, and memory blocks. Modular components can be swapped to explore new configurations or scheduling strategies quickly. FPGA implementations run end-to-end AI workloads under real hardware constraints. Integrated performance counters capture timing and power metrics directly on the FPGA.

### B. In-House Architecture: ADAM + Gemmini-SV

Our platform integrates ADAM, a parametric RISC-V microcontroller, with Gemmini-SV, a SystemVerilog-based variant of the Gemmini accelerator specialized for ML workloads. This combination creates a heterogeneous architecture composed of one or more RISC-V cores paired with TPU-like

systolic array accelerators. We can instantiate multiple ADAM cores, each attached to a Gemmini-SV instance with different systolic array sizes or memory technologies. For example, one core might use a small systolic array with a fast SRAM scratchpad, while another uses a larger array backed by a high-density emerging memory. At runtime, scheduling policies can choose the most advantageous configuration based on workload demands and memory-retention constraints.

### C. Software Stack and Scheduling

We leverage the IREE compiler and runtime to deploy generic ONNX models on our heterogeneous microcontroller platform. IREE's MLIR-based IR allows us to map operators to ADAM and Gemmini-SV and to orchestrate data movement. Our custom IREE backend implements scheduling heuristics at compile time, enabling experiments with operator tiling, data placement, and refresh or migration policies. By adjusting these policies, we can explore trade-offs between performance, energy, and retention overhead. Scheduling outcomes are validated on FPGA using performance counters and memory access logs.

### D. RTL Emulation of Emerging Memories

One issue with FPGA-based evaluations is the absence of physical emerging memory devices. To address this, we implement synthesizable RTL memory modules using FPGA BRAM to emulate novel cell behaviors. Each module includes configurable latency insertion and data-retention checks that trigger data invalidation after a set period. For example, our 2T model adds retention checks that mark data invalid when retention limits are exceeded and triggers an error on invalid data read. By embedding these RTL modules in Gemmini-SV's scratchpad, we measure cycle-accurate timing and energy overheads imposed by emerging memory constraints. Energy metrics can later be derived from memory access logs. This emulation-driven approach ties hardware behavior to software scheduling, enabling exploration of retention-aware design strategies.

## IV. Initial Results

Our FPGA-based exploration framework has achieved several key milestones toward enabling scheduling research. The ADAM microcontroller platform has been successfully deployed on FPGA, integrating a multiple RISC-V cores with standard peripherals (UART, GPIO, SPI, I2C, TIME). While operational, we are finalizing the non-intrusive monitoring unit that will enable precise metric collection without affecting execution behavior.

We reached a state with the Gemmini-SV accelerator where it can perform basic matrix operations ($R = A \cdot B + C$) with scratchpad data movement through its DMA engine. However, it is currently operating in a co-simulation between a RISC-V ISA emulator and the Gemmini-SV RTL simulation, with RTL-level integration to the ADAM platform in development.

Figure 1 shows our initial target architecture that combines ADAM with a single Gemmini-SV instance for scheduling experiments, with future scaling planned for multiple cores and accelerators.

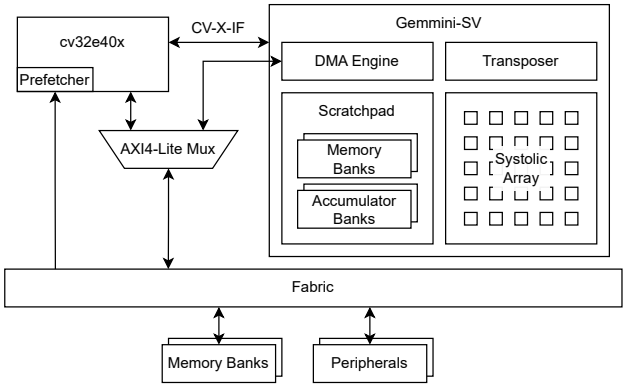



Fig. 1. Initial exploration architecture: an ADAM microcontroller instance with a single RISC-V core tightly coupled to a Gemmini-SV accelerator.

## V. Conclusion

In this proposal, we have outlined an end-to-end FPGA-based framework for studying the interaction between emerging on-chip memory technologies and heterogeneous AI accelerators. Our plan combines a parametric RISC-V microcontroller platform (ADAM), a customizable Gemmini-SV systolic accelerator, an MLIR-driven compiler stack, and RTL emulation of novel memory technologies to capture cycle-accurate performance, energy, and area trade-offs under limited-retention constraints. Implementation of this framework is ongoing: next steps include synthesizing the RTL memory modules in FPGA BRAM, completing the custom IREE backend, and validating scheduling heuristics on representative AI benchmarks. We will then extend our memory models to additional technologies (e.g., FeFET, MRAM) and refine data-migration and refresh strategies.

## References


[1] A. Symons, L. Mei, S. Colleman, P. Houshmand, S. Karl, and M. Verhelst, "Towards heterogeneous multi-core accelerators exploiting fine-grained scheduling of layer-fused deep neural networks," *arXiv preprint arXiv:2212.10612*, 2022, available: https://arxiv.org/abs/2212.10612.

[2] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, "Chipyard: Integrated design, simulation, and implementation framework for custom SoCs," *IEEE Micro*, 2020.

[3] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proceedings of the Annual Design Automation Conference (DAC)*, 2021.

[4] The IREE Authors, "IREE," Sep. 2019. [Online]. Available: https://github.com/iree-org/iree

[5] A. Lu, J. Lee, T.-H. Kim, M. A. U. Karim, R. S. Park, H. Simka, and S. Yu, "High-speed emerging memories for AI hardware accelerators," *Nature Reviews Electrical Engineering*, 2024.

# Design Optimization of CMOS-based Analog Spiking Neurons: A technological perspective

Harshit Kansal, Chhandak Mukherjee, and Cristell Maneux

***Abstract*—Minimizing power consumption in non-Von Neumann architectures necessitates in-memory computing, which can be effectively realized through Spiking Neural Networks (SNNs). CMOS-based analog circuits are fundamental to neuromorphic processors; however, transistor scaling introduces challenges due to short-channel effects (SCEs). This work explores the role of device architecture on the performance of CMOS-based analog neurons by comparing implementations based on a 28 nm Fully Depleted Silicon-on-Insulator (FDSOI) technology with that of a 28 nm Vertical Nanowire FET (VNWFET). Our results demonstrate that VNWFETs, due to their superior electrostatics, offer higher compactness thus requiring lower capacitance and can handle higher currents, making them promising candidates for energy-efficient, large-scale SNN implementations.**

***Index Terms*—Spiking Neural Network (SNN), Fully Depleted Silicon-on-insulator (FDSOI), Vertical Nanowire FET (VNWFET).**

## I. Introduction

ANALOG neurobiological circuits leverage non-linear behavior of analog components to enable adaptive information processing, offering distinct advantages over their digital counterparts [1]. One such implementation of an analog neuron for Spiking Neural Networks (SNNs) utilizes a one-transistor-one-resistor (1T1R) synapse, where the input neuron accumulates incoming spikes until reaching a threshold, at which point it generates an output spike. The number of synapses that can be accommodated before neuron firing—also known as the fan-in of the SNN—along with their distinction, strongly depends on the neuron's threshold. Several neuron circuit designs have been proposed in the literature [2], [3], with Fig. 1 illustrating a fundamental analog neuron design. As shown in Fig. 1a, where the input is a current, and Fig. 1b, where the input is a voltage, the latter model incorporates additional control mechanisms for the refractory period and output spike characteristics due to the presence of a higher number of transistors.

In this study, we adopt the neuron model depicted in Fig. 1a, implemented using both a 28 nm Fully Depleted Silicon-on-Insulator (FDSOI) technology and a 28 nm Vertical Nanowire FET (VNWFET). The VNWFET technology, in particular, offers enhanced immunity to short-channel effects (SCEs) [4]. A key advantage of VNWFETs in analog circuit design is their superior linearity, which is crucial for neuromorphic computations. Additionally, the vertical architecture of these transistors enables higher integration densities, facilitating compact and efficient circuit implementations. VNWFETs also exhibit low leakage currents, high ON/OFF ratios, and excellent control over threshold voltage variations, making them highly suitable for neuromorphic applications over FDSOI transistors.

H.Kansal, C.Mukherjee and C. Maneux are with the IMS Laboratory, University of Bordeaux, CNRS-UMR-5281, Bordeaux-INP, Talence, France, e-mail: harshit.kansal@u-bordeaux.fr.

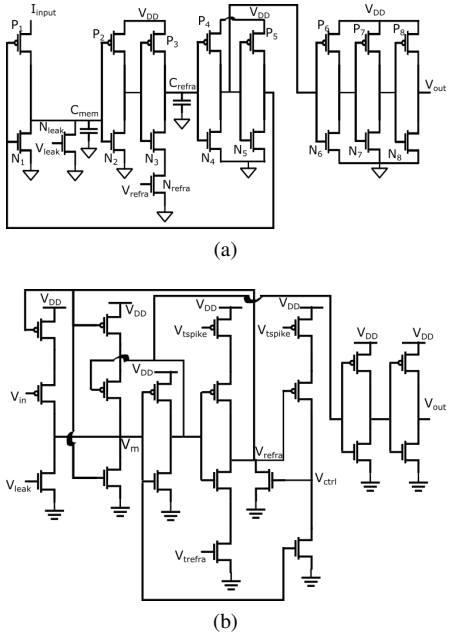



Fig. 1: 28 nm Analog Neuron Circuit for SNN, inspired from leaky-integrate and fire neuron, where integration happens at node $V_{mem}$ and capacitor $C_{mem}$ accumulates the incoming charge from the pre-synaptic neuron, while the controlled leakage is provided through a transistor with suitable $V_{leak}$.

In this simulation study, for an FDSOI device, we used the BSIM4 model along with the associated model card and included an appropriate netlist for neuron circuit simulations in the Analog Design System (ADS) environment. At the same time, the BSIM-CMG model was used for a VNWFET where the parasitic capacitances due to the source-to-gate ($C_{gs}$) and the drain-to-gate couplings ($C_{gd}$) were modified to adapt the netlist of the neuron circuit. The next section presents a detailed analysis of the neuron circuit based on these two implementations.

## II. Results and Conclusion

In this section, we first validate our circuit simulation results for the 28 nm FDSOI technology by ensuring that its behavior matches the findings reported in [2] (see Fig. 1a) and [3] (see Fig. 1b). This verification, presented in Fig. 2, allows

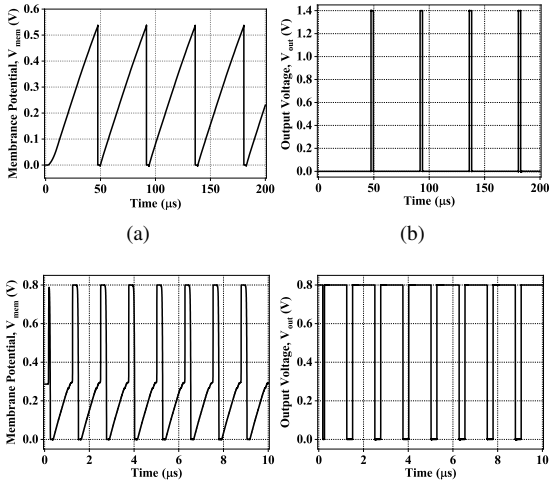



Fig. 2: **For the neuron circuit model in Fig. 1a (with FDSOI)** (a)$V_{mem}$ and (b) $V_{out}$, with an $I_{input}$ of 1.5 nA, $V_{DD}$ of 1.4 V, $V_{refra}$ of 0.14 V, $V_{leak}$ of 0.1 V, $C_{mem}$ of 82 fF and $C_{refra}$ of 8.6 fF. **For the neuron circuit model in Fig. 1b (with FDSOI)** (c) $V_{mem}$ and (d) $V_{out}$, with a constant $V_{in}$ of 0.4 V, $V_{DD}$ of 0.8 V, $V_{refra}$ of 0.2 V, $V_{leak}$ of 0 V and $C_{mem}$ of 1.44 fF. The voltage for controlling the width of the generated spikes, defined as $V_{tspike}$, is kept at 0.35 V.

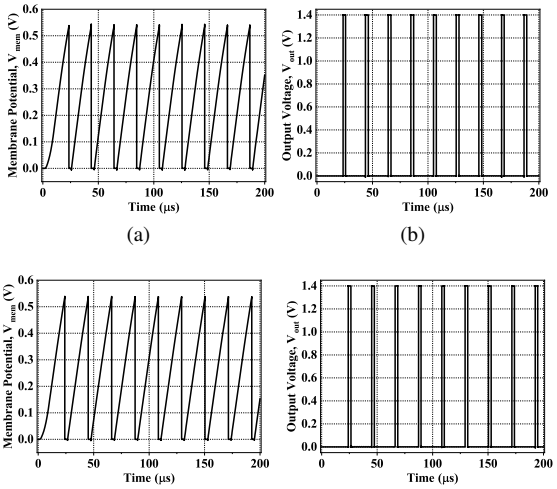



Fig. 3: **For the neuron circuit model in Fig. 1a (with FDSOI)**: (a)$V_{mem}$ and (b) $V_{out}$, with an $I_{input}$ of 1.5 nA and $C_{mem}$ **of 28 fF** . (c)$V_{mem}$ and (d) $V_{out}$, with an $I_{input}$ **of 3 nA** and $C_{mem}$ of 82 fF.

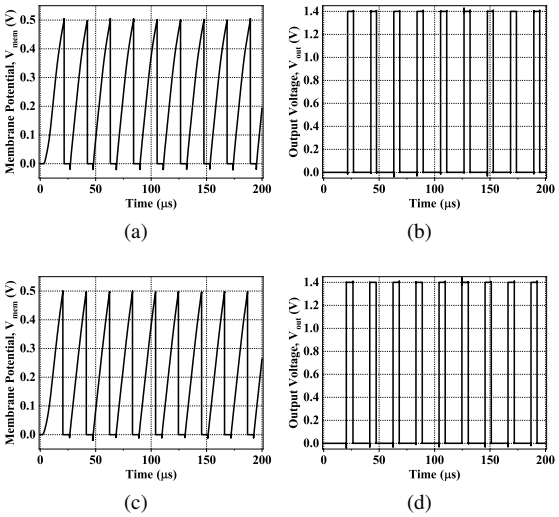



Fig. 4: **For the neuron circuit model in Fig. 1a (with VNWFET)**: (a)$V_{mem}$ and (b) $V_{out}$, with an $I_{input}$ of 1.5 nA and $C_{mem}$ **of 10 fF** . (c)$V_{mem}$ and (d) $V_{out}$, with an $I_{input}$ **of 3.2 nA** and $C_{mem}$ of 60 fF.

us to analyze both circuits and identify the input current amplitude ($I_{input}$) and the membrane capacitance ($C_{mem}$) as key parameters. Figure 3 illustrates the impact of $I_{input}$ and $C_{mem}$ on $V_{mem}$ and $V_{out}$. A comparison with Fig. 1a and Fig. 1b reveals that decreasing $C_{mem}$ or increasing $I_{input}$ leads to a higher number of output spikes, highlighting the limiting values for these parameters. Building on this understanding, we compare the results in Fig. 3 with those obtained using the VNWFET-based circuit (Fig. 4). By ensuring an identical number of output spikes in both FDSOI- and VNWFET-based implementations, we observe that due to the superior electrostatics of VNWFETs, the circuit requires a smaller $C_{mem}$ of 10 fF for the VNWFET compared to the value of 28 fF for the FDSOI architecture, while accommodating a higher input current of 3.2 nA for the VNWFET compared to that of 3 nA for its FDSOI counterpart.

In conclusion, VNWFET-based circuits demonstrate significantly better energy consumption and area economy, making them highly suitable for large-scale spiking neural networks. In future work, our aim is to extend this study to larger networks with a more precise VNWFET model.

## ACKNOWLEDGMENT

The authors thank the IMS Laboratory for financial and infrastructure support.

## REFERENCES

[1] Indiveri, G. and Liu, S.C., "Memory and information processing in neuromorphic systems", *Proceedings of the IEEE*, Vol.103, No.8, pp.1379-1397, 2015.

[2] Palhares, J.H.Q., Beilliard, Y., Sandrini, J., Arnaud, F., Garello, K., Prenat, G., Anghel, L., Alibart, F., Drouin, D. and Galy, P., "A tunable and versatile 28 nm FD-SOI crossbar output circuit for low power analog SNN inference with eNVM synapses", *Solid-State Electronics*, Vol. 209, p.108779, 2023.

[3] Cincon, V., Vatajelu, E.I., Anghel, L. and Galy, P., "From 1.8 V to 0.19 V voltage bias on analog spiking neuron in 28nm UTBB FD-SOI technology", *IEEE Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS)*, Pages 1-4, 2020.

[4] Guerfi, Y. and Larrieu, G., "Vertical silicon nanowire field effect transistors with nanoscale gate-all-around", *Nanoscale research letters*, Vol. 11, pp.1-7, 2016.

# Mapping a DNN Model to Heterogeneous Hardware Resources

Salsabil Saoudi, Mario Barbareschi, Alberto Bosio
*Ecole Centrale de Lyon, INL, UMR5270, 69130 Ecully, France*
{firstname.lastname}@ec-lyon.fr, mario.barbareschi@unina.it

***Abstract*—Efficient deployment of deep neural networks (DNNs) on resource-constrained hardware is increasingly challenging due to growing model complexity. This work presents a framework that partitions DNNs across multiple systolic array configurations, all operating under the same dataflow strategy. Performance evaluation is guided by MAESTRO, an analytical cost model that captures data reuse and estimates key metrics such as latency and energy. Based on the performance metrics, the framework optimizes partitioning and hardware allocation. An Integer Linear Programming (ILP) optimization algorithm is then used to explore the design space and select the optimal hardware configuration for the target trained DNN model.**

***Index Terms*—Deep Neural Network, Design Space Exploration, Partitioning, MAESTRO, ILP, Systolic Array, Optimization Algorithm**

## I. Introduction

A typical DNN model consists of numerous neurons and a large set of parameters, known as model weights or coefficients. These are learned during the training phase using the backpropagation algorithm combined with gradient descent.

Once the DNN is trained sufficiently with adequate data, it can be used for inference. In this phase, the model applies its learned weights to new input data to generate predictions for tasks such as image classification or object detection. However, due to the large computational demand of deep networks, it becomes essential to explore methods that can optimize performance and reduce latency without impacting the accuracy of the model. One such approach is model partitioning, which has already shown promise in training scenarios and becomes even more interesting in inference contexts, where the frequency of execution is typically much higher than training.

Partitioning a DNN requires dividing the model into smaller, computationally manageable components. These partitions can be strategically distributed across multiple hardware resources, thereby optimizing resource utilization and enhancing computational efficiency in distributed environments. To enable such deployment, two principal partitioning strategies are commonly adopted: vertical partitioning, which divides the model along its layers, and horizontal partitioning, which splits computations within layers across devices. Additionally, pipelining can be applied to further enhance computational efficiency by overlapping the execution of different partitions. This technique allows multiple stages of the DNN computation to be processed simultaneously, reducing overall inference latency and improving throughput.

Although distributing, deploying, and executing large DNN models on multiple, potentially heterogeneous hardware resources is a promising approach, it currently requires significant manual effort. This involves advanced skills in DNN model design, embedded system programming, and parallel programming for heterogeneous distributed systems. Currently, there is no comprehensive design and programming framework that can fully automate the partitioning and deployment of a trained DNN model across multiple hardware resources. In the long term, the framework we propose aims to address these challenges. By automating the partitioning, optimization, and hardware allocation process, our framework will significantly reduce manual design effort, making distributed DNN on hardware more accessible and efficient.

We propose a framework that partitions DNN model layers across multiple systolic array configurations using a shared dataflow. Performance is evaluated using the MAESTRO [1] analytical cost model, which analyzes data reuse and provides key metrics such as latency and energy. Based on MAESTRO's results, the framework optimizes partitioning and hardware allocation. An Integer Linear Programming (ILP) approach is then used to determine the optimal hardware configuration for efficient DNN deployment.

## II. Methodology

The framework aims to explore how DNNs can be deployed more efficiently across specialized hardware. The idea stems from the observation that DNNs often require significant computational resources, and intelligently distributing the workload across multiple systolic array configurations could lead to meaningful improvements in performance and efficiency. In this initial work, I focus on partitioning the DNN model into smaller segments, each mapped to a distinct systolic array configuration, while maintaining a consistent dataflow strategy throughout the system.

To guide the partitioning process and evaluate hardware performance, we integrate MAESTRO, as illustrated in Figure 1, a state-of-the-art analytical cost model specifically designed for systolic array-based DNN accelerators. MAESTRO takes as input a DNN model description, a data-centric mapping representation, and hardware resource specifications. These specifications include key architectural parameters such as the number of Processing Elements (PEs), global buffer size, and local buffer size, as depicted in Figure 2. By analyzing different forms of data reuse within the accelerator, MAESTRO esti-

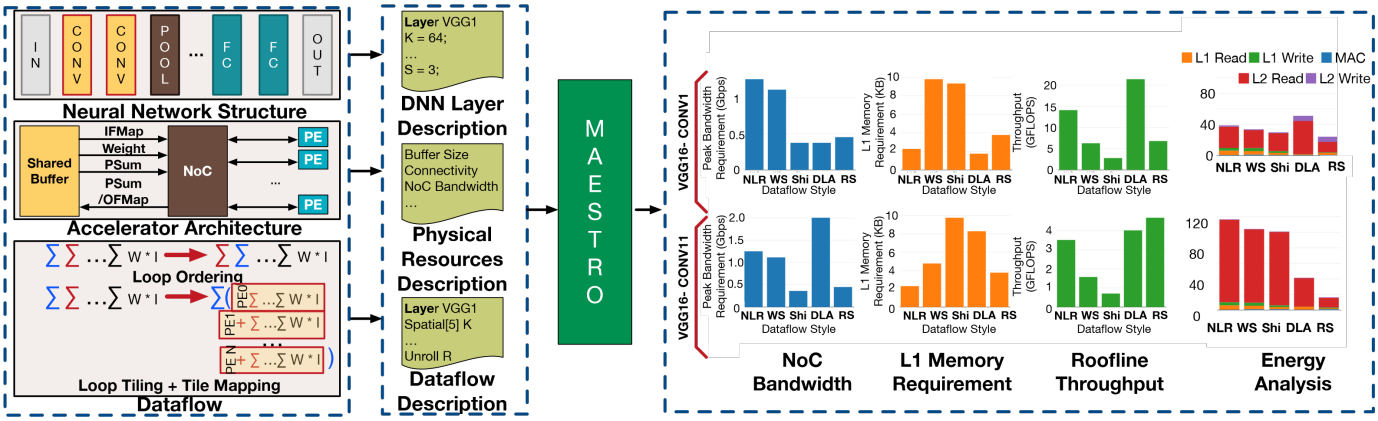



Fig. 1. Overview of MAESTRO: Analytical cost model for DNN dataflows [2].

mates over 20 performance metrics, including latency, energy consumption, throughput, and area, offering critical insights for optimizing hardware design and partitioning strategies.

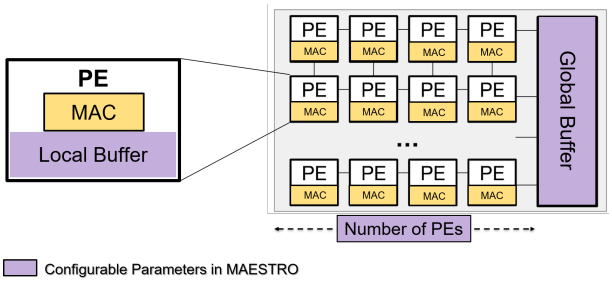



Fig. 2. Overview of systolic array

Based on the output from MAESTRO, we attempt to optimize the use of the hardware by adjusting the partitioning strategy to better align with the computational characteristics of each DNN layer. To identify the most efficient configuration among the possible options, I formulate the problem as an ILP model as explained in this multi-objective optimization function:

$$\min\left(\alpha \cdot \text{Latency} + \beta \cdot \text{Energy}\right) \tag{1}$$

where $\alpha$, and $\beta$ are weighting factors. Taken into account these constraints:

$$\sum_{j} x_{i,j} = 1 \quad \forall i \quad ; \quad \text{Area} < \gamma \tag{2}$$

where $x_{i,j}$ layer i assigned to hardware configuration j and $\gamma$ is weighting factor.This allows to systematically explore the design space and select the optimal hardware set-up based on defined performance goals.

Figure 3 illustrates the results of the framework, which presents a layer-wise latency and energy consumption for four different hardware configurations of sysolic array applied to the same DNN model. These configurations have the same number of PEs and vary in key architectural parameters, such as the global buffer size and local buffer size, as outlined above.

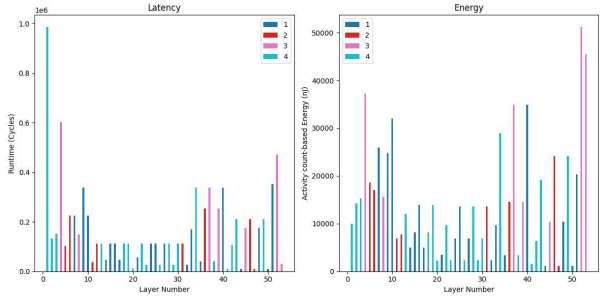



Fig. 3. Latency and Energy of the best Hardware Configuration using MAESTRO for MobileNetV2 model

As shown, both performance metrics exhibit significant variation between layers, highlighting the sensitivity of different layers to hardware resource allocation. In particular, deeper layers tend to be more compute-intensive, leading to higher runtime and energy.

## III. Conclusion and Future Work

This work presented a framework for efficient DNN deployment by partitioning models across multiple systolic arrays under the same dataflow, guided by the MAESTRO cost model and optimized via ILP. The approach improves inference efficiency and simplifies hardware allocation. Future directions include analyzing data movement between system components, partitioning across heterogeneous hardware (CPUs, GPUs, TPUs), adapting to diverse dataflows, and evaluating hardware accelerator reliability to further improve deployment robustness and performance.

## References

[1] H. Kwon, P. Chatarasi, V. Sarkar, T. Krishna, M. Pellauer, and A. Parashar, "Maestro: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings," *IEEE micro*, vol. 40, no. 3, pp. 20–29, 2020.

[2] H. Kwon, P. Chatarasi, M. Pellauer, A. Parashar, V. Sarkar, and T. Krishna, "Understanding reuse, performance, and hardware cost of dnn dataflow: A data-centric approach," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 754–768, 2019.