IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

SPICE-Level Demonstration of Unsupervised Learning with Spintronic Synapses in Spiking Neural Networks

SALAH DADDINOUNOU[1], ANTENEH GEBREGIORGIS[2], SAID HAMDIOUI[2], ELENA-IOANA VATAJELU[1]
[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France
[2]Department of Quantum and Computer Engineering, Delft University of Technology, Delft, The Netherlands

Corresponding author: Elena-Ioana Vatajelu (e-mail: ioana.vatajelu@univ-grenoble-alpes.fr).

This work is funded by the Agnece Nationale de la Recherche - France within the EMINENT Project ANR-19-CE24-0001

ABSTRACT Spiking Neural Networks (SNNs) are Artificial Neural Networks which promise to mimic the biological brain processing with unsupervised online learning capability for various cognitive tasks. However, SNN hardware implementation with online learning support is not trivial and might prove highly inefficient. This paper proposes an energy-efficient hardware implementation for SNN synapses. The implementation is based on parallel-connected Magnetic Tunnel Junction (MTJ) devices and exploits their inherent stochasticity. In addition, it uses a dedicated unsupervised learning rule based on optimized Spike-Timing-Dependent Plasticity (STDP). To facilitate the design of the SNN, its training and evaluation, an open-source Python-based platform is developed; it takes as input the SNN parameters and discrete circuit components, and it automatically generates the associated full netlist in SPICE; moreover, it extracts the simulation results and makes them available in python for evaluation and manipulation. Unlike conventional neuromorphic hardware that relies on simple weight mapping post-off-line training, our approach emphasizes continuous, unsupervised learning, ensuring an energy efficiency of 11.2nW per synaptic update during training and as low as 109fJ/spike during inference.

INDEX TERMS MTJ, Neuromorphic, SNN, STDP, unsupervised learning.

I. INTRODUCTION

ARTIFICIAL Intelligence (AI) is transforming both society and the economy with two crucial drivers at its core: *energy-efficient* computing and *unsupervised online* learning [1][2]. These qualities are particularly important for IoT and edge computing devices which must operate on limited battery power and adapt in real-time. Current cloud-based AI chips (being power-hungry and relying on offline training) have large silicon footprints, suffer from static power and bandwidth constraints due to memory bottleneck [3][4][5]. To address these issues, AI must not only achieve up to 100 times more energy efficiency but also facilitate post-deployment continuous unsupervised learning. *Brain-inspired* computing, especially *Spiking Neural Networks (SNNs)* [6][7] that emulate the brain's Spike-Timing Dependent Plasticity (STDP), have shown a huge potential in achieving both energy-efficient and unsupervised computing. However, the development of low-power, reliable hardware for SNNs remains an open question for research.

The work published on the hardware implementation of SNN can be classified into three classes, based on their computing paradigm. The first class is based on computing platforms such as CPUs, GPU and TPUs, which make use of the traditional Von-Neumann architecture and are capable of handling intense parallel computing[8][9][10]. Although these platforms deliver high accuracy, they consume significantly more power compared to other neuromorphic architectures. The second class are near-memory implementations, focusing on power efficiency, they include dedicated CMOS-based computing engines such as TrueNorth[11] and Loihi [12]. However, these chips, despite their near-memory character, still suffer from memory bottleneck which limits the bandwidth and requires massive data shuffling. The third class uses in-memory computing paradigm, which makes it more biologically plausible. The synapses that hold the weights are stored in non-volatile (NV) technology devices[13][14] in a crossbar array between the neurons. However, the vast majority of the aforementioned proposals don't support online learning[15]; they only map the trained weights to the crossbar array for inference, after training the network offline. Although there are a few works that proposed the use of Magnetic Tunnel Junctions (MTJs) for plasticity dynamics in SNN[16][17][18], there is a need to demonstrate this in full network training. In short, the state-of-the-art of SNN implementations suffer from high power consumption, poor biological plausibility, and are unable to perform unsupervised online learning in a full SNN. Hence, there is a need for new energy-efficient implementations of SNNs, that are capable of learning in an unsupervised manner, and do this online, which means continuous learning after deployment.

This study advances the state-of-the-art of SNN hardware implementation by introducing *energy-efficient* spintronic synapses, combined with an *adapted STDP* learning rule[19], to perform *unsupervised learning*, and develops a *user-friendly framework* for SNN training in SPICE. In short, the major contributions of the paper are:

- Introduction of **multistate synapses** by exploiting the inherent stochasticity of parallel connected MTJs.
- Implementation of an **optimized STDP rule** tailored to the unique characteristics of our spintronic synapses.
- Proposition of **behavioral neuron models** for both input (spiking) and output (integrate-and-fire) neurons.
- Development of an **automated Python framework** [1] interfacing between user-defined parameters and SPICE simulator, featuring netlist generation, training and evaluation.
- **SPICE validation** of unsupervised SNN training followed by an evaluation on a functional framework demonstrated an accuracy above 90% on MNIST data recognition.
- Demonstration of a very **energy-efficient** SNN with **11.2nW** per synaptic update during training and as low as **109fJ/spike** during inference.

The remainder of the paper is organized as follows: Section II provides some background about MTJs and SNNs. Section III details the contributions of this article including the synapse structure, the learning rule, and the neurons' models, section IV explains the modules of the the developed SNN framework. Next, section V shows the obtained results with discussion. Finally, section VI concludes the paper.

II. BACKGROUND

A. MAGNETIC TUNNEL JUNCTION (MTJ)

A Magnetic Tunnel Junction is a nanoscale structure comprising two ferromagnetic layers separated by an insulating oxide barrier Fig. 1B. The thicker ferromagnetic layer, often referred to as the "reference" layer (RL), has a fixed magnetization direction. In contrast, the thinner layer, known as the "free" layer (FL), has a magnetization that can be manipulated to be either parallel or antiparallel to the reference layer. The relative orientation of these layers determines the MTJ's resistance: a parallel configuration results in low resistance (R_{low} or "0"), while an anti-parallel configuration yields high resistance (R_{high} or "1"). The MTJ's functionality is harnessed in Spin Transfer Torque Random Access Memory (STT-RAM) applications, where the bistable resistance states represent binary data. The transition between these states is driven by Spin Transfer Torque (STT), a phenomenon wherein the angular momentum from spin-polarized electrons influences the magnetization direction of the free layer. Our simulations employ a comprehensive VerilogA MTJ model [20], which encompasses a wide range of parameters, such as geometric dimensions, saturation magnetization, damping factor, and crystalline anisotropy. This model uses the Landau-Lifshitz-Gilbert (LLG) equation [21], a fundamental equation in magnetism that describes the temporal evolution of magnetization under various influences:

$$\frac{\partial \vec{m}}{\partial t} = -\gamma\mu_0\vec{m} \times \vec{H}_{\text{eff}} + \alpha\vec{m} \times \frac{\partial \vec{m}}{\partial t} - \beta J\vec{m} \times (\vec{m} \times \vec{m}_r) \quad (1)$$

where $\vec{m}$ represents the unit magnetic moment of the FL magnetization under the macrospin approximation. $\vec{H}_{\text{eff}}$ is the effective magnetic field, which is the sum of different magnetic fields. γ is the gyromagnetic ratio and μ_0 is the vacuum permeability. α is the Gilbert damping constant. β is the STT coefficient. J is the switching current density, and $\vec{m}_r$ is the unit vector of the RL magnetization.

The switching dynamics of the MTJ, governed by STT, are inherently probabilistic due to thermal fluctuations. The model distinguishes two regimes based on the magnitude and duration of the switching current: the Sun model (for currents greater than the critical current, $I > I_{c0}$) and the Neel-Brown model (for $I < 0.8I_{c0}$). The former is characterized by rapid switching events but at the cost of higher power consumption, while the latter -used in our simulations- exhibits slower, thermally-assisted switching with reduced power consumption.

B. SPIKING NEURAL NETWORKS

The brain comprises billions of neurons interconnected by trillions of synapses [22]. This network is unique in that the communication inter-neuron is carried through spikes, which confers a low power consumption to the brain. SNNs that mimic the biological brain are inspired by this mode of communication but also the method of learning that comes with it. Biological neurons integrate incoming spike voltages from other connected neurons, forming the membrane potential. When this potential surpasses a firing threshold, the neuron emits a spike, ensuring information is conveyed only upon reaching this threshold. Various models, such as the Hodgkin and Huxley model and the Leaky Integrate and Fire (LIF) model, have been developed to emulate biological neurons, some modules focus on the biological plausibility and others on the simulation speed [23] [24]. Since the SNN communicates only through spikes, encoding the datasets that need to be fed to the network as spikes is another important topic of investigation. This encoding can be achieved through

[1]**The entire framework will be made Open Source upon acceptance.**

methods like rate, frequency, or temporal coding. Another active topic of research is SNN training, many works in the literature propose training the SNN in two steps, first training a usual ANN with backpropagation, then, once the training is finished, the ANN is converted into an SNN for inference [25] but both training and inference run in classical hardware. Unfortunately, this does not fully unlock the potential of SNN in terms of energy efficiency. We find also a slightly better option in the literature, which consists of training an SNN in classical hardware and then mapping the weights into a cross-bar array of memristors [26]. Although this method is better because it uses In-memory computing for inference, the memristors of the network don't learn online. Our method is more biologically plausible because we train the network directly in the crossbar array of MTJ-based synapses. Indeed, the NV synapses in the crossbar change their stored weights as the network is learning according to STDP learning rule. An SNN consists of input spiking neurons (presynaptic) and output spiking neurons (postsynaptic), interconnected by synapses. These synapses can be arranged in a crossbar array as in Fig. 1A, with input and output neurons positioned at the ends of each row and column, respectively. Communication between neurons is facilitated by spike trains. In tasks like image recognition using frequency coding, each pixel is associated with one input neuron (or three input neurons for colored images), with the neuron's spiking frequency proportional to the pixel's intensity. Hardware-implemented SNNs with probabilistic MTJ synapses is explored. In such architectures, the synaptic weight is represented in conductance levels, and a local training algorithm based on STDP is employed.

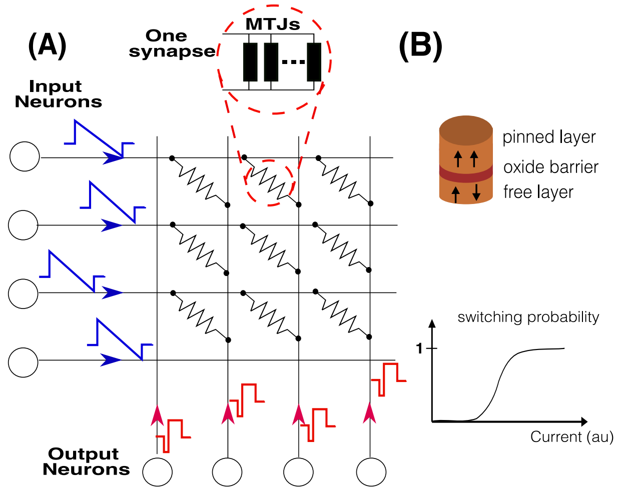


FIGURE 1: **(A)** A schematic of a fully connected SNN composed of: 1) a crossbar array of spintronic synapses, each synapse is a set of multiple MTJs in parallel. 2) input neurons that generate a train of triangular voltage shapes. 3) output neurons that fire a bi-rectangular voltage shape upon crossing the membrane threshold. **(B)** Top: A schematic of the MTJ structure. Bottom: Probabilistic switching of the MTJ.

III. PROPOSED SYNAPSE, NEURONS AND ASSOCIATED LEARNING RULE

The goal of our study is to demonstrate unsupervised training of SNN in hardware. To this end, we don't content with simulating the high-level functionalities of the synapses and neurons, instead, we run electrical simulations with accurate device models that capture the physics of the devices. In this section, we present the overall architecture of the SNN. We start by presenting the device models required to run SPICE simulation, we present the spintronic synapses, we describe the input and output neurons, and we explain the learning dynamics. Subsequently, we show how important it is to automate the design process, especially for large SNNs. Finally, we introduce our Python framework that allows for netlist automatic generation and its interaction with the Specter simulator.

A. MTJ-BASED SYNAPSE

The SNN under study employs synapses, where each is a compound of multiple MTJs connected in parallel. Table 1 displays the MTJ parameters that are used in simulations. The resulting equivalent conductance of the synapse could take one of multiple states. Indeed, a single-MTJ device inherently supports only two distinct conductance states. However, by integrating multiple MTJs in parallel, a broader range of equivalent conductance levels can be achieved. The number of possible states of conductance in the synapse is equal to N+1, where N is the number of MTJs. In this design, we operate the MTJ devices in the stochastic regime to facilitate online training and reduce power consumption. When a certain writing voltage is applied, the synapse will end up in one of the possible states with a certain probability. This is because the switching of state in each MTJ is probabilistic, and the final state after writing depends on how many MTJs are in parallel and how many are in an anti-parallel configuration.

TABLE 1: MTJ Parameters

Parameter	Description	Value
α	Gilbert Damping Coefficient	0.027
P	Electron Polarization Percentage %	52
H_k	Out of plane Magnetic Anisotropy (Oe)	1433
M_s	Saturation Field in the Free Layer (Oe)	15800
r	Radius of the MTJ nanopillar (nm)	16
t_{sl}	Height of the Free Layer (nm)	1.3
t_{ox}	Height of the Oxide Barrier (nm)	0.85
TMR	TMR(0) with Zero Volt Bias Voltage %	70
T	Temperature (K)	300
RA	Resistance area product ($\Omega\mu m^2$)	5

B. LEARNING RULE

Our objective is to leverage the probabilistic behavior of the synapse so that when it receives signals from the pre-and post-synaptic neurons, its conductance will be updated according to a customized STDP rule, the Bi-sigmoid STDP, presented in detail in [19]. On one hand, we know that the probability that the spintronic synapse switches from one state to another depends only on the magnitude and duration of the voltage

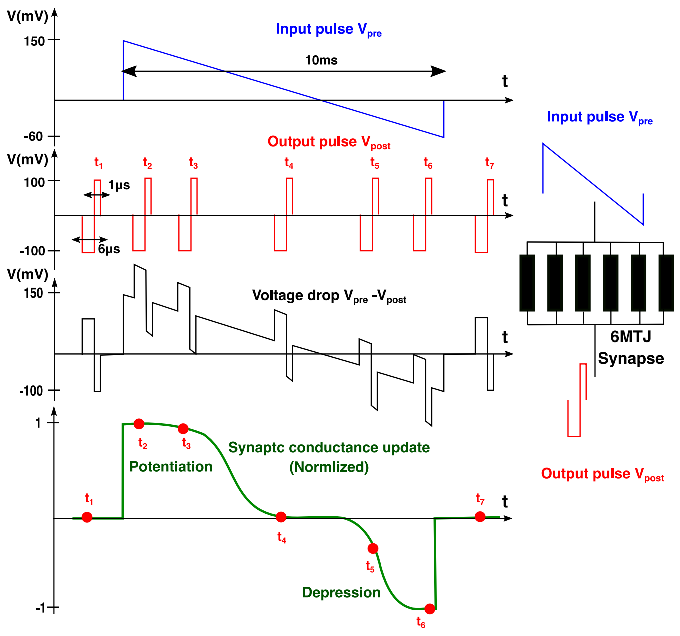


FIGURE 2: Temporal relationship between pre-synaptic input pulses (V_{pre}), post-synaptic output pulses (V_{post}), and the resultant synaptic weight modifications. The first two panels depict V_{pre} and a sequence of V_{post} respectively.The third panel shows the resulting voltage drop across the synapse ($V_{pre} - V_{post}$). The bottom panel provides a normalized view of the synaptic conductance update, highlighting periods of potentiation and depression corresponding to the timing sequences t1 through t7. The inset on the right shows a 6MTJ synapse.

drop across its terminals. On the other hand, in STDP, the crucial parameter that determines the value of the weight update is the relative time between input and output neuron spiking. To reproduce STDP, we manipulate the probability of switching by tuning the voltage drop across the synapse, so that an increase of conductance (potentiation) takes place when the input-output relative spiking time is small. Likewise, a decrease in conductance (depression) takes place when the input-output relative spiking time is large. This switching probability manipulation can be easily obtained by custom-designing the voltage responses of input and output neurons. A thorough investigation -that extends beyond the scope of this article- allowed us to find the optimal voltage shapes of input and output neuron signals. A triangular shape for the input neuron voltage (blue pulse in Fig. 2), and a double square voltage shape for the output neuron (red pulse in Fig. 2). The weight update happens when the output pulse overlaps (in time) with a part of the input pulse, the resulting voltage drop across the synapse $V_{\text{pre}} - V_{\text{post}}$ depends on the relative time between spikings. This voltage drop which alters the probability of switching is shown in Fig. 2 alongside the input and output voltage profiles. Depending on the relative spiking time between input and output, The obtained STDP (green curve in Fig. 2) can be divided in five regions:

1) **High Potentiation:** Immediate post-synaptic spiking after pre-synaptic results in a maximum $V_{\text{pre}} - V_{\text{post}} > 0$, significantly increasing the synapse conductance.
2) **Low Potentiation:** A smaller positive voltage drop with increasing delay slightly raises the synapse conductance.
3) **Unchanged Conductance:** A transitional phase between potentiation and depression where the voltage drop diminishes and cannot induce either potentiation or depression.
4) **Low Depression:** A substantial delay yields a negative $V_{\text{pre}} - V_{\text{post}}$, reducing the synapse's conductance.
5) **High Depression:** A significant delay between pre- and post-synaptic pulses results in a large negative voltage drop, drastically decreasing the synapse conductance.

C. NEURON MODELS

1) Input neuron

We focus on the black-and-white image recognition task. The number of input neurons corresponds to the image's pixel count. Each input neuron, modeled in Verilog-A, translates the intensity of a single pixel into a series of triangular voltage pulses (first panel of Fig. 2). Crucially, the frequency of these pulses is directly proportional to the pixel's intensity, establishing a clear intensity-to-frequency conversion mechanism. A small challenge arises from neurons associated with pixels of zero intensity, as the proportional conversion would not generate any pulse, thus preventing that neuron from contributing to learning. To address this, a small frequency bias is introduced to the spike trains of all input neurons. This ensures that even zero-intensity pixels contribute to the synaptic training process. Fig. 3 presents a schematic of the input neuron and how the designed voltage shape could be generated. As the input neuron model is part of a bigger framework, further customization is possible through several adjustable parameters accessible via the framework's interface. This includes the ratio for converting intensity to frequency, the duration for which an input image is presented to the network, and the specific characteristics of the triangular voltage pulses. These adjustable parameters allow for fine-tuning of the input neuron model to suit various image recognition scenarios, enhancing the system's flexibility and efficacy in processing diverse visual inputs.

2) Output neuron

The behavioral model of the output neuron describes a leaky integrate-and-fire (LIF) neuron adapted from [27]. This LIF neuron which is depicted in Fig. 4 integrates incoming signals and produces a double rectangular spike when its membrane potential surpasses a predefined threshold. At its core, the LIF circuit operates as an RC (Resistor-Capacitor) circuit, it is composed of a capacitor, a charging, and discharging resistances, and finally, a voltage source that generates the double rectangular spike. Three switches control the signal flow in the neuron giving place to three primary phases: integration, leakage, and spiking. During the integration phase,

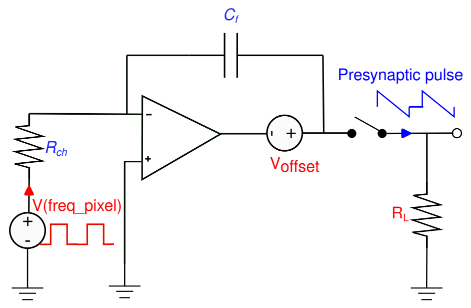


FIGURE 3: Presynaptic neuron design, generating the specific presynaptic pulses with a proportional frequency to the pixel intensity. The design is described by a behavioral model.

only the charging switch is connected to charge the capacitor with incoming synaptic signals. If no incoming spike is being received, leakage starts with connecting another switch that allows the capacitor to discharge via Rdischarge while the charging switch is disconnected. When the voltage across the capacitor reaches the predefined threshold of the membrane potential, the neuron enters the spiking phase, during which a third switch connects the neuron terminal to a voltage source that generates back the postsynaptic signal to update the synapse connections. During this time, the charging switch is disconnected and the discharging switch is connected to a high resistance (Rpostdischarge) for a fast resetting of the membrane potential.

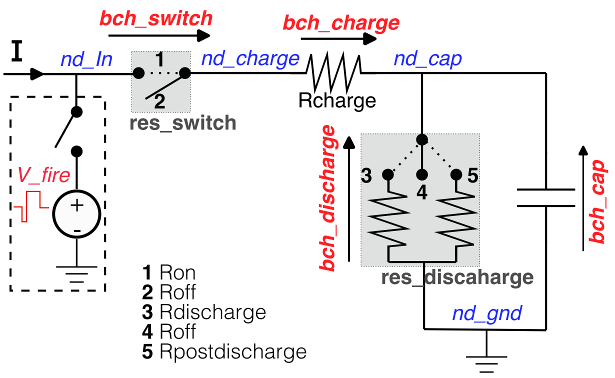


FIGURE 4: Postsynaptic neuron design described by a behavioral model

IV. AUTOMATED SNN FRAMEWORK

Development and training of SNNs at the device level presents some unique challenges due to the extensive computational resources required. Our *open-source* framework addresses these challenges by automating the design and training of SNNs for electrical simulations, specifically tailored for SPICE environments.

♣ Framework Overview:

Our tool depicted in Fig. 5 streamlines the creation of intricate SNN architectures for Spice simulations. It facilitates the generation of detailed netlists, which are essential for accurately modeling the neural network's behavior at the device level. Unlike general netlist generators, such as those referenced in [28] and [29], our tool is specifically optimized for SNN applications. It incorporates user-defined parameters, including the number of input and output neurons, as well as the configuration of MTJs in each synapse, thereby allowing for a high degree of customization.

♣ Integration with Spice Simulator:

A notable feature of the framework is its seamless integration with Cadence Ocean tool. This compatibility enables users to easily set simulation parameters all at once, the tool then takes care of updating all the necessary files accordingly. Our tool automates the simulation process, including the recording of various signals for subsequent analysis.

♣ Simulation and Analysis:

Post-simulation, the framework presents comprehensive results that encompass synaptic weight evolution, neuron membrane potential dynamics, and overall learning performance. This analysis is conducted in the background, requiring minimal user input beyond the initial setup.

♣ Parallel Processing Capabilities:

One of the core strengths of the framework is its multiprocessing capability while satisfying the simulator constraints. This feature is particularly beneficial for conducting parallel simulations and parametric analysis across different SNN designs, significantly reducing the time and computational resources required for extensive exploratory studies.

♣ Key Modules:

The framework's architecture which is summarized in Fig. 5 is modular, with the main Python script (`snn_simulator.py`) coordinating the flow between the following components:

- **Parameter Specification:** Design choices and simulation options are respectively handed to `netlist_generator.py` and `run_SPICE.py` from the main script.
- **Netlist Generation:** `netlist_generator.py` dynamically constructs SNN netlists, adaptable to various sizes and architectures, and outputs to `netlist`.
- **Spice Simulation Execution:** `run_SPICE.py` integrates with Cadence Ocean scripts (`oceanScript.ocn`) to automate simulation processes and output to designated folders.
- **Performance Analysis:** Leverages `visualisation.py` to display key signals, facilitating an analysis of SNN training efficacy.
- **Parallel Processing:** `snn_simulator.py` integrates multiprocessing with parametric analysis to simultaneously conduct multiple simulations for large design exploration.

Each module is geared towards simplifying and automating the complex processes involved in SNN design, training, and analysis. The main script orchestrates the entire simulation process. It initializes with user-defined parameters, generates the netlist through a dynamic and scalable approach, and interfaces with simulation tools for execution. The use of Python's multiprocessing capabilities not only accelerates the simulation process but also allows for the simultaneous exploration of multiple SNN configurations.

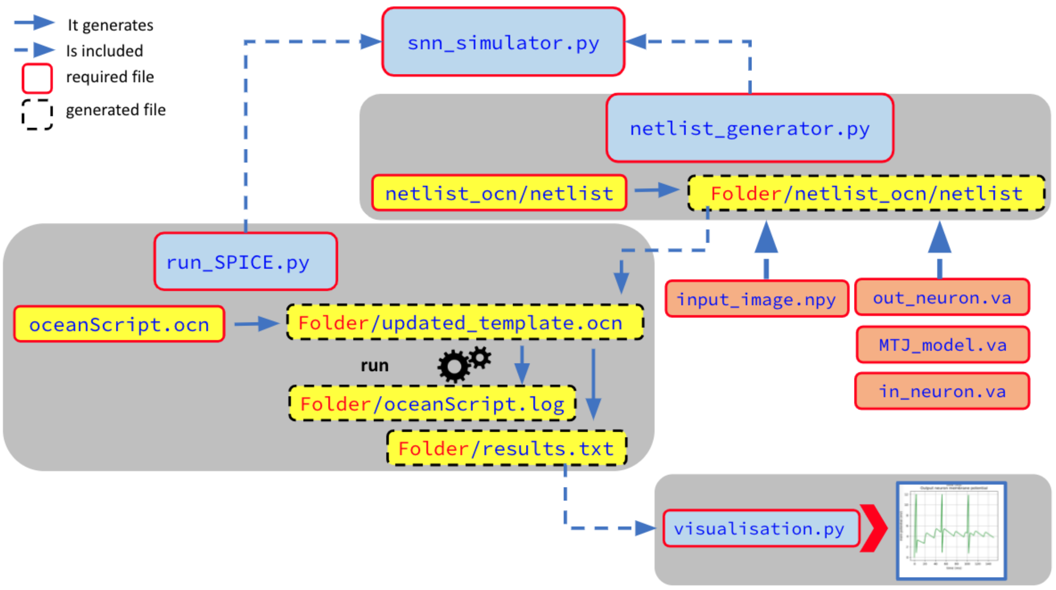


FIGURE 5: Overview of the SNN Training Framework detailing module interactions and data flow.

V. RESULTS AND DISCUSSION

In this section, we first demonstrate the efficiency of the proposed framework performing online training. SPICE simulations emulate the SNN training process to recognize a small dataset selected merely as an illustrative example. The derived leaning rule is then evaluated in a larger SNN using a functional simulator. We then evaluate the power efficiency of training and inference, and show the effect of spiking activity and the synapse composition in the energy consumption.

A. VALIDATION WITH THE PROPOSED SPICE FRAMEWORK

Our approach to training SNNs is unique because it *trains online* directly on the MTJ crossbar array. Unlike typical methods where an external algorithm trains the SNN conventionally before mapping the weights to the crossbar, our system updates connections in response to the ongoing activity of input and output signals, without relying on a separate, explicit algorithm. However, the valuable demonstration of hardware-based SNN training in SPICE comes at the cost of dealing with computational overhead, which is influenced by the number of input and output neurons, and the number of MTJs per synapse. Each component adds more currents and voltages to be calculated. Given the intense computational demands and lengthy processing times, we choose to illustrate our approach in a manageable way, using 25 input neurons. Each neuron is associated with a single pixel of an input image that represents a character in a 5x5 pixel format. For computational considerations, we let the SNN learn one character at a time, hence only one output neuron is required. Our main objective through this basic SNN is to show that our design together with the customized STDP online learning rule, performs effectively while maintaining reasonable simulation times. We trained the SNN to learn 10 characters shown in Fig. 6, where each couple of images shows the synaptic conductances before and after training. Initially, conductances of the synapses are randomly initialized, then after 150 ms of presenting the image to the SNN, the synapses perfectly learn that character. It is worth noting that the whole process happens with no supervision, but solely thanks to the input-output activity that updates the MTJ-based synapses accordingly.

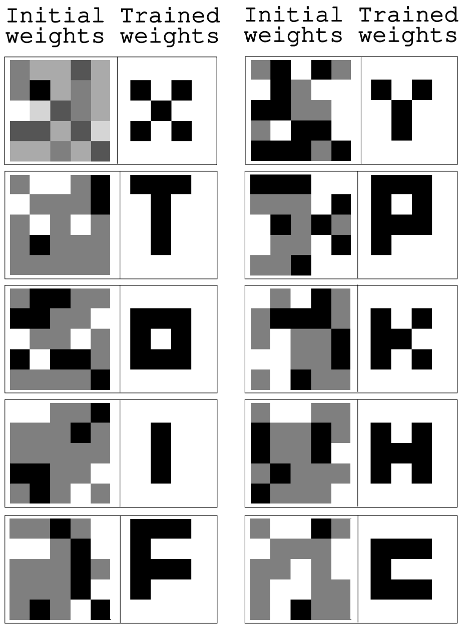


FIGURE 6: Synaptic weights before and after training, 10 showcases are presented. The weights are randomly initialized and finish by learning the presented character each time.

We first trained the network to learn the character "X" (top left on Fig. 6) in a network that has 6MTJs per synapse, when the network perfectly learned, we reduced the number of MTJs to only 2MTJs per synapse for all other trainings. As shown in Fig. 6, SNN always learns the presented character, which indicates the robustness and versatility of our design. The Fig. 7A provides a temporal evolution of synaptic weights dynamics when the network is learning the character "X". First, the 25 synapses are distributed randomly through 7 states, then as learning evolves, some weights potentiate and others depress until stabilization. The Fig. 7B illustrates the membrane potential of the output neuron over time, capturing the neuron's accumulation, leakage, and firing activity in response to incoming spikes.

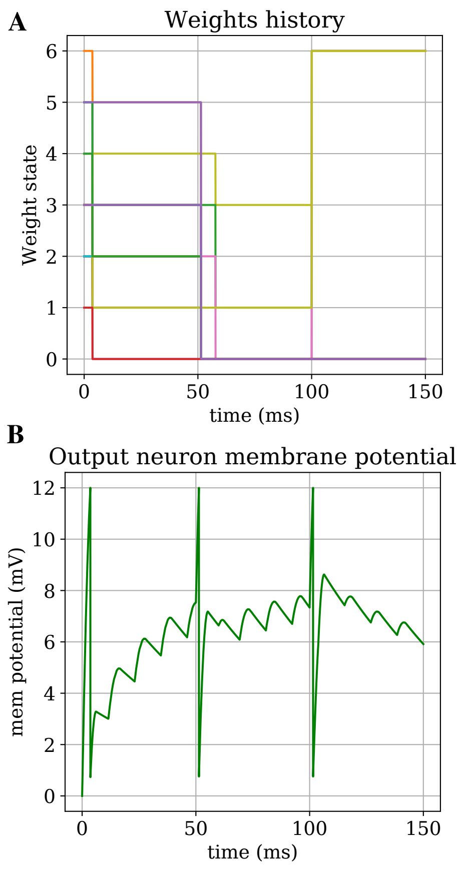


FIGURE 7: **(A)** Synaptic weight history during the network's training phase, showing transitions between various conductance states. **(B)** Membrane potential dynamics of the output neuron, depicting the threshold-triggered firing and subsequent weight adjustment in the synapses.

B. VALIDATION WITH A FUNCTIONAL FRAMEWORK

In our study, we first utilized SPICE simulations to authenticate the efficacy and physical compatibility of our newly derived STDP learning rule with MTJ-based synapses in a small dataset. This approach confirms that our learning rule is both theoretically robust and directly applicable to neuromorphic hardware, leveraging MTJ's advantages for efficient in-memory computing. Given SPICE's detailed electrical simulations, which align closely with the equations governing the synaptic devices, we could validate our STDP rule's device-level feasibility. However, to explore the scalability and broader applicability of our learning rule, we injected our STDP in Bindsnet [30], a functional framework to speed up the simulations, while ensuring that it uses a learning rule that originated from SPICE-based simulations of the spintronic synapse. This shift allowed us to apply our learning rule to train an SNN on MNIST dataset [31] with the architecture presented on [32] adopting leaky integrate-and-fire neuron models and conductance-based synapses, and consisting of 784 input neurons and 400 output neurons. The Fig. 8 illustrates the network's classification performance on the MNIST dataset over three epochs, showing a progressive improvement as it processes an increasing number of training samples. After training, the network was evaluated, achieving an accuracy of 90.28%. While this figure may be modest compared to state-of-the-art ANNs, it's crucial to highlight the significance of this achievement within the context of SNNs. The observed performance underscores not only the energy efficiency inherent to SNNs but also the advantage of their unsupervised learning nature, as opposed to the supervised paradigm common in DNNs. Moreover, the foundation of our learning rule in the physical properties of MTJ-based synapses positions it as a particularly fitting choice for neuromorphic computing, aligning closely with the operational principles of neuromorphic hardware.

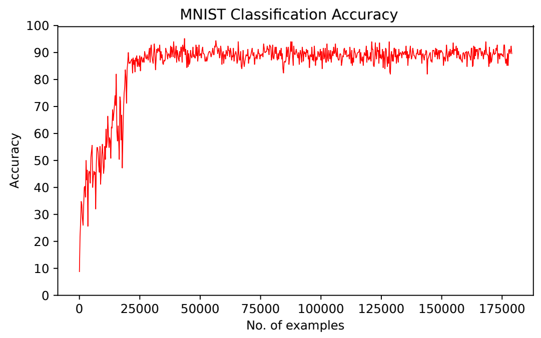


FIGURE 8: Incremental improvement in classification performance as a function of the number of training samples.

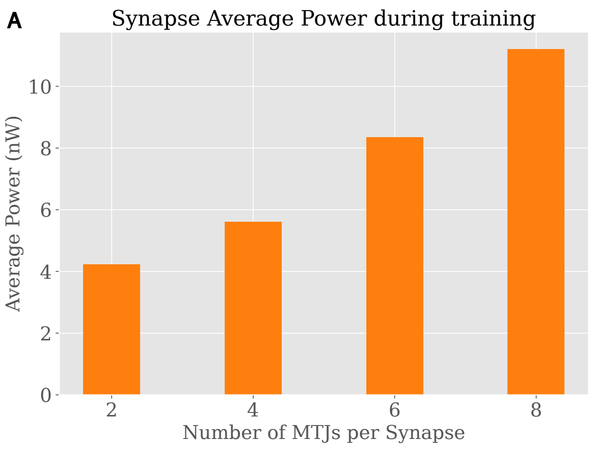
C. ENERGY EFFICIENCY

Employing multiple MTJs in parallel as synapses in SNNs presents a significant advantage in reducing energy consumption. We analyzed the power consumption of synapses (excluding the power consumption of the neurons and of the rest of the system). The synapses are equipped with varying numbers of MTJs (2 to 8) depending on design choices which are dictated by the complexity of the learned task and the needed resolution of the synapse. Our analysis focuses also on the power consumption dynamics of the synapses when subjected to different spiking frequencies (20Hz to 80Hz). This range of frequencies corresponds to the encoding of black and white pixels, respectively. Notably, during inference phases, by disabling feedback from the output neuron—which exhibits high impedance—we achieve a significant reduction in power consumption. The power estimation of a single synapse is quantified through SPICE simulations, which capture the average power $\overline{P}$ over a 150ms image presentation interval, calculated as:

$$\overline{P} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} P(t)\, dt \tag{2}$$

Where $t_1 - t_0 = 150ms$. Averaging the power as given in Eq. 2 is performed to account for the varying instantaneous power within a given spike, but also for the silent inter spike intervals. The energy per spike is finally derived by multiplying the average power by the spike's duration (set at $10ms$ in this analysis). Notably, the power consumption escalates with the number of MTJs per synapse due to a decrease in the equivalent resistance of the synapse. This effect is depicted in Fig. 9A which shows the power consumption of synapses with different number of MTJs during training at a maximum spiking frequency of $80Hz$. Consequently, our analysis benchmarks in Fig. 9B the upper bound of power consumption using 8-MTJ synapses, which are subjected to varying neuron spiking frequencies. The results show a power consumption ranging from $1.29nW$ to $11.2nW$ corresponding to an energy consumption per spike of $112pJ$ and $12.9pJ$ for synapses connected to the most and least active neurons, spiking at $80Hz$ and $20Hz$ respectively, during training. Inference stages show even more promising figures, with energy per spike dropping to as low as $438fJ$ and $109fJ$ for synapses connected to the most and least active neurons, respectively. This efficient synapse is neuron-agnostic and can be matched with any neuron design available on the literature. This allows more optimization from the neuron side for the energy efficiency of the system.

In comparing various SNN implementations, several works rely on offline training, followed by weights mapping to a crossbar array of synapses based either on Resistive Random Access Memory (RRAM) [33] [26] or MTJ [34], the network is then intended for inference without online learning capabilities. The works in [33] and [35] employed gradient descent algorithm (supervised) and reported inference energy efficiencies of $3.6fJ$/spike experimentally and $270fJ$/spike in SPICE, respectively. However, the offline nature of their training limits real-time application potential. In a similar approach, [26] trains an SNN using unsupervised learning with STDP, yet still follows an offline training methodology before mapping the weights to the RRAM crossbar for SPICE simulations, resulting in $20fJ$/spike energy consumption. The authors of [34] demonstrated that Spin-Orbit Torque (SOT) MTJs are very energy efficient for inference. The work demonstrates experimentally the plasticity of a compound synapse composed of 16 SOT MTJ devices. the MNIST offline training of a network followed by weight mapping to SOT-MTJ based crossbar array achieved $1.3fJ$



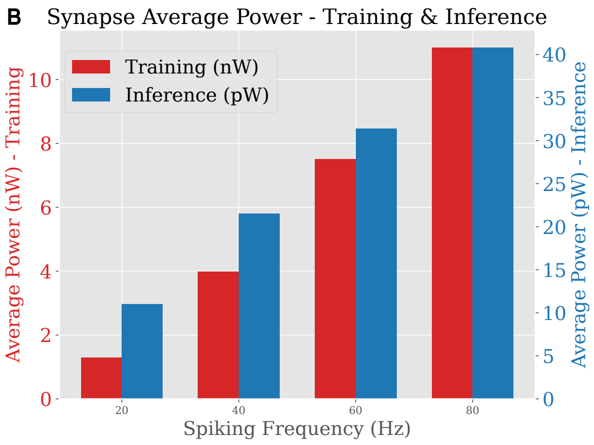


FIGURE 9: Comparison of power consumption of the synapse. **(A)** Average power consumption during training with different counts of MTJ per synapse at a fixed spiking frequency of 80 Hz. **(B)** Average power consumption during training (nW) and inference (pW) across varying spiking frequencies, in 8-MTJs-based synapses.

per spike during inference. Meanwhile, comprehensive evaluations of power consumption of training within networks using unsupervised STDP on MTJ-based synapses, especially via accurate SPICE simulations, remain limited. Furthermore, the variation in communicated metrics, including total power consumption, energy per spike, and energy per image, complicates the task of establishing clear benchmarks, hence we choose to benchmark the power required by a basic operation during training which is a single weight update. The study by [36] introduces a simplified STDP mechanism for synapses based on a single MTJ, specifically targeting vehicle counting applications. The authors preferred using a local fast simulator over the accurate but slow SPICE simulations. The reported weight update power dissipation ranges from $180nW$ to $0.42mW$ depending on the spiking mode. In this context, the study by [37] explores stochastic computing by performing system-level simulations of networks using synapses comprised of MTJs in series, with configurations extending to as many as 49 MTJs per synapse. Their simplified STDP involves three phases in one training process: communication, potentiation, and depression. For a synapse configuration of 4 MTJs in series, the authors reported an estimated power consumption of 47.6 µW to update the state of the synapse. Table 2 compares our work to other proposals.

TABLE 2: Training method & energy efficiency comparison

Network	**Training**	**Learning rule**	**Weight update power**	**Energy per spike (Inference)**
El Arrasi[26]	Offline	Unsupervised (STDP)	N/A	20 fJ
Valentian [33]	Offline	Supervised (backprop)	N/A	3.6 pJ
Ostwal [34]	Offline	Supervised (backprop)	N/A	1.3 fJ
Zhang[37]	Online	Unsupervised (simplified STDP)	47.6 µW	–
Vincent [36]	Online	Unsupervised (simplified STDP)	180 nW	–
This work	Online	Unsupervised (Dynamic STDP)	11.2 nW	109 fJ

Even with our most consuming synapse composed of 8 MTJs and subject to the highest spiking frequency (80Hz), the power dissipation of our proposed synapse during training outperforms both the proposals of [37] and [36]. Unlike the mentioned proposals, our work uses a dynamic STDP that properly interprets the relative time between spikes of pre- and post-synaptic neurons and updates the spintronic weights accordingly. During inference, our proposal is more energy efficient than [33] but it is surpassed by [26] and [34]. However, our work not only uses unsupervised STDP adapted to the physics of spintronic synapses, but it also implements it in an online manner, which is not the case of [26] and [34]. This feature, we believe, is more suited for IoT devices and real-time applications, offering a balance between energy efficiency and practical applicability in dynamic environments. Our work's nuanced approach to synapse design and online training not only reduces power consumption but also ensures compatibility across various neuron models, fostering greater optimization opportunities for overall system energy efficiency.

VI. CONCLUSION

In this paper, we presented an energy-efficient implementation of SNN. Our approach enhances SNNs for unsupervised online learning by utilizing a specialized STDP learning rule that is rooted in the physics of MTJ-based synapses and exploits the inherent stochasticity of MTJs. An open-source Python platform developed for this purpose simplifies SNN training and evaluation, automating SPICE netlist generation and simulation. The results showcase our design's effectiveness, achieving an energy efficiency of 11.2nW per synaptic update during training and surpassing conventional neuromorphic hardware that relies on post-offline training weight mapping. These results pave the way for future advancements in autonomous systems, IoT devices, and edge computing.

References

[1] Adelson Chua, Michael I Jordan, and Rikky Muller. "SOUL: An energy-efficient unsupervised online learning seizure detection classifier". In: *IEEE Journal of Solid-State Circuits* 57.8 (2022), pp. 2532–2544.

[2] Rachmad Vidya Wicaksana Putra and Muhammad Shafique. "Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE. 2021, pp. 1057–1062.

[3] Le Ye et al. "The challenges and emerging technologies for low-power artificial intelligence IoT systems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.12 (2021), pp. 4821–4834.

[4] Hiroshi Momose, Tatsuya Kaneko, and Tetsuya Asai. "Systems and circuits for AI chips and their trends". In: *Japanese Journal of Applied Physics* 59.5 (2020), p. 050502.

[5] Yufei Ma et al. "In-memory computing: The next-generation ai computing paradigm". In: *Proceedings of the 2020 on Great Lakes Symposium on VLSI*. 2020, pp. 265–270.

[6] Wolfgang Maass. "Networks of spiking neurons: the third generation of neural network models". In: *Neural networks* 10.9 (1997), pp. 1659–1671.

[7] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "Third generation neural networks: Spiking neural networks". In: *Advances in computational intelligence*. Springer, 2009, pp. 167–178.

[8] Toru Baji. "GPU: the biggest key processor for AI and parallel processing". In: *Photomask Japan 2017: XXIV Symposium on Photomask and Next-Generation Lithography Mask Technology*. Vol. 10454. SPIE. 2017, pp. 24–29.

[9] Yuxin Wang et al. "Benchmarking the performance and energy efficiency of AI accelerators for AI training". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE. 2020, pp. 744–751.

[10] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. "Benchmarking TPU, GPU, and CPU platforms for deep learning". In: *arXiv preprint arXiv:1907.10701* (2019).

[11] Paul A Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface". In: *Science* 345.6197 (2014), pp. 668–673.

[12] Mike Davies et al. "Loihi: A neuromorphic manycore processor with on-chip learning". In: *Ieee Micro* 38.1 (2018), pp. 82–99.

[13] Gabriel Maranhão and Janaina Gonçalves Guimarães. "Low-power hybrid memristor-CMOS spiking neuromorphic STDP learning system". In: *IET Circuits, Devices & Systems* 15.3 (2021), pp. 237–250.

[14] Aijaz H Lone, Selma Amara, and Hossein Fariborzi. "Magnetic tunnel junction based implementation of spike time dependent plasticity learning for pattern recognition". In: *Neuromorphic Computing and Engineering* 2.2 (2022), p. 024003.

[15] Deming Zhang et al. "All spin artificial neural networks based on compound spintronic synapse and neuron". In: *IEEE transactions on biomedical circuits and systems* 10.4 (2016), pp. 828–836.

[16] Salah Daddinounou and Elena Ioana Vatajelu. "Synaptic control for hardware implementation of spike timing dependent plasticity". In: *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. IEEE. 2022, pp. 106–111.

[17] Yunho Jang et al. "Stochastic SOT device based SNN architecture for On-chip Unsupervised STDP Learning". In: *IEEE Transactions on Computers* 71.9 (2021), pp. 2022–2035.

[18] Thomas Leonard et al. "Shape-Dependent Multi-Weight Magnetic Artificial Synapses for Neuromorphic Computing". In: *Advanced Electronic Materials* 8.12 (2022), p. 2200563.

[19] Salah Daddinounou Elena Ioana Vatajelu. "Bi-Sigmoid Spike-Timing Dependent Plasticity Learning Rule for Magnetic Tunnel Junction-based SNN". In: *Front. Neurosci. Sec. Neuromorphic Engineering* 18 (2024).

[20] Yue Zhang et al. "Compact model of subvolume MTJ and its design application at nanoscale technology nodes". In: *IEEE Transactions on Electron Devices* 62.6 (2015), pp. 2048–2055.

[21] Yuan Xu, Shuai Wang, and Ke Xia. "Spin-transfer torques in antiferromagnetic metals from first principles". In: *Physical review letters* 100.22 (2008), p. 226602.

[22] Clarence Tan, Marko Šarlija, and Nikola Kasabov. "Spiking neural networks: background, recent development and the NeuCube architecture". In: *Neural Processing Letters* 52 (2020), pp. 1675–1701.

[23] Katrin Amunts et al. "The human brain project: creating a European research infrastructure to decode the human brain". In: *Neuron* 92.3 (2016), pp. 574–581.

[24] Eugene M Izhikevich. "Which model to use for cortical spiking neurons?" In: *IEEE transactions on neural networks* 15.5 (2004), pp. 1063–1070.

[25] Rivu Midya et al. "Artificial neural network (ANN) to spiking neural network (SNN) converters based on diffusive memristors". In: *Advanced Electronic Materials* 5.9 (2019), p. 1900060.

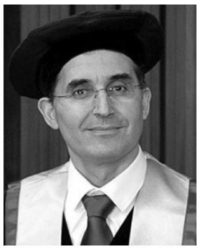
[26] Asmae El Arrassi et al. "Energy-efficient SNN implementation using RRAM-based computation in-memory (CIM)". In: *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE. 2022, pp. 1–6.

[27] Sumedha Gandharava Dahl. "The Effects of Radiation on Memristor-Based Electronic Spiking Neural Networks". In: (2020).

[28] Sergio Vinagrero Gutiérrez, Giorgio Di Natale, and Elena-Ioana Vatajelu. "Python Framework for Modular and Parametric SPICE Netlists Generation". In: *arXiv preprint arXiv:2306.12224* (2023).
[29] PySpice Contributors. *PySpice: Python Bindings for NgSpice and Tools to Support SPICE Simulation*. https://github.com/PySpice-org/PySpice. GitHub repository. 2023.
[30] Hananel Hazan et al. "BindsNET: A Machine Learning-Oriented Spiking Neural Networks Library in Python". In: *Frontiers in Neuroinformatics* 12 (2018), p. 89. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00089. URL: https://www.frontiersin.org/article/10.3389/fninf.2018.00089.
[31] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems* 2 (1989).
[32] Peter U Diehl and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity". In: *Frontiers in computational neuroscience* 9 (2015), p. 99.
[33] A Valentian et al. "Fully integrated spiking neural network with analog neurons and RRAM synapses". In: *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE. 2019, pp. 14–3.
[34] Vaibhav Ostwal et al. "A novel compound synapse using probabilistic spin–orbit-torque switching for MTJ-based deep neural networks". In: *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 5.2 (2019), pp. 182–187.
[35] Donguk Lee et al. "Various threshold switching devices for integrate and fire neuron applications". In: *Advanced Electronic Materials* 5.9 (2019), p. 1800866.
[36] Adrien F Vincent et al. "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems". In: *IEEE transactions on biomedical circuits and systems* 9.2 (2015), pp. 166–174.
[37] Deming Zhang et al. "Energy-efficient neuromorphic computation based on compound spin synapse with stochastic learning". In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2015, pp. 1538–1541.

SALAH DADDINOUNOU is currently a PhD student at TIMA laboratory 38000, Grenoble. His PhD thesis focuses on the robustness of hardware implementations of bio-inspired neural networks (Spiking Neural Networks) by using emerging technologies (memristors and/or spintronic devices).

ANTENEH GEBREGIORGIS (Member, IEEE) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2019. He is currently a Postdoctoral Researcher with the Computer Engineering Laboratory, Delft University of Technology, Delft, The Netherlands. His research interests include emerging technologies, artificial intelligence (AI), computation in-memory, neuromorphic computing, architectures for ultra-low power design, reliability analysis, and variability assessment of VLSI devices.



SAID HAMDIOUI (Senior Member, IEEE) received the M.S.E.E. and Ph.D. degrees (Hons.) from the Delft University of Technology (TU Delft), The Netherlands. Before joining TU Delft as a Professor, he was with Intel, Santa Clara, CA, USA, Philips Semiconductors Research and Development, Crolles, France, and Philips/NXP Semiconductors, Nijmegen, The Netherlands. He is currently a Chair Professor of dependable and emerging computer technologies and the Head of the Computer Engineering Laboratory (CE-Lab), TU Delft. He owns two patents, has published one book and contributed to the other two, and had coauthored more than 250 conference papers and journal articles. He has consulted for many semiconductor companies in the area of memory testing. His research interests include dependable CMOS nanocomputing (including reliability, testability, and hardware security) and emerging technologies and computing paradigms (including 3-D stacked ICs, memristors for logic and storage, and in-memory computing). He was a recipient of many international/national awards, such as the Best Tech Idea Award of The Netherlands, in 2021, the European Commission Components and Systems Innovation Award for the most innovative H2020 Project, the 2015 HiPEAC Technology Transfer Award, and the European Design Automation Association Outstanding Dissertation Award, in 2003. He received many best paper awards and nominations at leading international conferences, such as DATE 2020 and 2021, ITC 2021, ETS 2021, IVSLSI 2016, HPCS 2016, and ICCD 2015). He was an Associate Editor of many journals, including IEEE Transactions on Very Large Scale Integration (VLSI) Systems and JETTA. He serves on the editorial board of IEEE Design & Test and ACM Journal on Emerging Technologies in Computing Systems (JETC).

ELENA-IOANA VATAJELU (Member, IEEE) is full-time researcher with CNRS on the design, test and reliability of Integrated Circuits at TIMA Laboratory in Grenoble, France. She obtained her PhD from Polytechnic University of Catalunya Spain in 2011. Her expertise is on the reliability and the robustness assessment, design-for-reliability, test strategies and security primitives for CMOS and beyond CMOS RAMs in traditional and non-Von Neumann computing paradigms.