Analog Spiking Neural Network Synthesis for the MNIST

Thomas Soupizet, Zalfa Jouni, Siqi Wang, A. Benlarbi-Delai, Pietro Maris Ferreira

Analog Spiking Neural Network Synthesis for the MNIST

Thomas Soupizet, Zalfa Jouni, Siqi Wang, Aziz Benlarbi-Delai, Pietro M. Ferreira
Université Paris-Saclay, CentraleSupélec, CNRS, Lab. de Génie Électrique et Électronique de Paris,91192,Gif-sur-Yvette,France.
Sorbonne Université, CNRS, Lab. de Génie Électrique et Électronique de Paris, 75252, Paris, France.
email: soupizett@gmail.com, zalfa@ieee.org, siqi.wang, aziz.benlarbi_delai@sorbonne-universite.fr, maris@ieee.org

***Abstract*— Different from classical artificial neural network which processes digital data, the spiking neural network (SNN) processes spike trains, and hence can bring new advantages to the information processing. Indeed, its event-driven property helps to capture the rich dynamics the neurons have within the brain, and the sparsity of collected spikes helps reducing computational power. Besides, the nonlinear behavior of neurons offers easier solutions for non-linearity problems than classical methods and is well suited to stochastic resonance-based systems. An analog SNN composed of 86 electronic neurons (eNeuron) and 1238 synapses interacting through two hidden layers is proposed. It is made from two different types of eNeurons based on different versions of Leaky integrate-and-fire (LIF) or Morris-Lecar (ML) models. The proposed neural network, coupling deep learning and ultra-low power, is trained using a common machine learning system (TensorFlow) for the MNIST. LIF eNeurons implementations present some limitations in terms of dynamic range, while, considering different activation functions, ML eNeurons achieve robust accuracy which is approximately of 0.82.**

***Index Terms*— neuromorphic computing, spiking neural network, deep learning, MNIST**

I. Introduction

Big data has been for many years the cornerstone of machine learning in data science field. Such data acquisition is made on energy limited systems such as edge devices, Internet-of-Things (IoT), and later uploaded on the Cloud. It is on the Cloud, where classic Artificial Intelligence (AI) does its best treating such as big data at the expense of dozens of kW. Moreover, uploading such a big data has unaffordable cost of around dozens of μJ per bit for most IoT devices running from 16 to 64 bits applications. Such IoT devices are often powered by Lithium-ion batteries charged with a hundred Ampere-per-hour energy. The challenge in smart IoT is achieving a sub nano-Joule per sample while running edge-AI tasks to minimize the data transmitted to the Cloud. A smart IoT device then becomes a neuromorphic computing device capable of mimicking biological systems in order to achieve a better energy efficiency (E_{eff}) than digital systems when performing certain cognitive tasks [1].

Feed-forward neural networks (FNN) presents interneuron connections with a linear scaling controlled by the weight coefficients (synapses). FNNs have been dominant in software AI research aiming at deep learning capabilities with interested mathematical network properties [2]. Previous study of mathematical FNNs reveals necessary conditions for deep learning using spiking neural networks (SNN). SNN hardware aims to emulate biological systems in artificial neurons and synapses (electronic circuitry). They have often been proposed in FPGAs [3] and GPUs [4] due to its advantages of reconfigurability, reusability and reduced implementation costs. Analog systems are often chosen as they can faithfully mimic biological systems in artificial neurons and synapses. Figure 1 illustrates a model often used for spiking eNeurons and synapse for analog implementations. Analog eNeurons have presented the best energy consumption per unit of information (E_{eff} in J/spike) and a competitive area trade-off. Besides, a mixed circuit approach has been explored for a better trade-off [5].

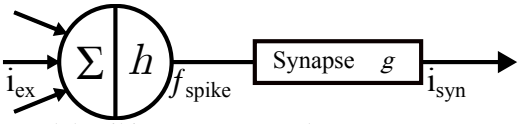


Fig. 1: Model spiking eNeuron and synapse. A neuron converts a current input to a spiking frequency f_{spike}. Synapse converts a spiking input to an output current.

A previous paper [6] has founded a study of deep neural network feasibility using ultra-low-power eNeuron. There, previous published electronic neurons (eNeurons) using the (Leaky) Integrate-and-Fire (LIF) [7, 8], and Morris Lecar (ML) simplified (simp.) or biomimetic (bio.) [9, 10] models have been redesigned using in-house BiCMOS 55 nm technology from ST Microelectronics.While mathematical properties of analog spiking neural networks are rarely a priority, energy efficiency (E_{eff}, in fJ/spike) of advanced software neural network architectures is rarely studied [11]. The tools used for software neural networks could be more useful in implementing deep learning for analog neural networks such as [12]. Thus, this paper proposes to bridge the gap between software AI and analog neural network research subjects. Further study of mathematical FNNs revealed set necessary conditions for deep learning using spiking neural networks. Thus, deep learning and energy efficiency are mutually exclusive if those neuromorphic components are used [6].

This paper innovates with a framework capable of evaluating analog SNN training and inference. Accurate analog SNN, illustrated in this work, is composed of 86 eNeurons and 1238 synapses with two hidden layers. This neural network is trained by taking advantage of the widely used machine learning framework Tensorflow [13]. It operates at large scale and in heterogeneous environment while supporting a variety of training and optimization algorithms. In this case, Adam optimization is chosen to obtain an efficient and scalable neural network [14]. The proposed analog SNN synthesis framework is illustrated for LIF, simp. and bio. ML eNeurons. Novel synthesis framework has highlighted the trade off revealed in [6]. Moreover, this work supplies an algorithm guiding designers into deep learning and energy-efficient analog SNN using MNIST.

Digital Object Identifier 10.29292/jics.v18i1.3

This paper is organized as follows. Section II. reviews literature from learning and inference point of view for neural networks implemented in either software or hardware, while presenting common building blocks used in analog SNN. Section III. proposes a neural network synthesis framework considering the MNIST to achieve a general solution for SNN implementation on analog hardware. Section IV. illustrates the post-layout simulation (PLS) results of the proposed technique for analog SNN training and inference. Finally, conclusions are drawn towards deep learning neural networks using ultra-low power eNeurons.

II. Background

Literature review is presented firstly from training and inference point of view for neural networks implemented in either software or hardware (i.e. digital design) in Subsec. II.A.. Secondly, state-of-the-art of common building blocks used in analog spiking neural networks is revised in Subsec. II.B. and Subsec II.C.. Comparisons in terms of often presented figures of merit (FoM) in eNeurons and synapses are detailed.

A. Neural Network training on MNIST

Machine learning has gained significant interest facing the tremendous increase of data collection in recent years. Researchers have focused on building learning algorithms to improve performances in a wide variety of applications from engineering sciences to the social network and security [15]. Neuromorphic architectures appears to be the most appropriate platform to implement machine learning algorithms due to their capability to learn and process as biological brain [11]. Diverse learning algorithms and models are demonstrated in literature. Designs of neuron, synapse and neural network models have an impact on the choice of the algorithm. Some training is implemented off chip and later transferred to a neural network, while others are based on the real-time learning mechanisms [1].

One commonly used dataset for testing in machine learning field is the Modified National Institute of Standard and Technology dataset (MNIST) [16]. MNIST is a large database of square 28*28-pixel gray-scaled picture of handwritten digits (from 0 to 9). The MNIST database contains a total of $70,000$ instances, where $60,000$ are usually used for training and the remainder are used for inference. During training step, images are fed to neural network architecture and synaptic weights are updated. During inference step, images are fed to a trained network to assess its accuracy in data recognition. Many learning algorithms are made from the standard MNIST benchmark optimal design networks capable of treating data recognition of handwritten digits and image processing systems.

Convolutional neural networks (CNNs) are the most well known and widely used techniques for the MNIST [15]. Wan et al. [17] have used CNNs using methods such as dropout training and reported accuracy as high as 99.79% with data augmentation. A topology called HyperNEAT is applied to the MNIST in [18], to support the evolution of CNNs by adding new substrate capable of representing this type of network. Their work led to the accuracy of 92.1%. Baldominos et al. [19] achieves the accuracy of 99.63% using grammatical evolution. Many other works in the modern machine learning algorithms present very high accuracy proposals [15]. However, none of them prove a plausible implementation in hardware.

The large-scale neuromorphic systems have presented chip designs in digital [1]. A spiking backpropagation algorithm based on synfire-gated dynamic information is implemented in Loihi neuromorphic processor [20]. Loihi's network is composed of 810 neurons with two hidden layers. It achieves the accuracy of 95.7% after 60 epochs on the MNIST. Processing a single sample consumes 0.592 mJ of energy on neuromorphic cores.

Esser et al. [21] demonstrates the accuracy of 92.7% for a trained network using the backpropagation operating in a probabilistic domain. The network runs on the TrueNorth digital chip using the MNIST. The proposed network has 5 cores distributed in 2 layers, which corresponds to 512 neurons. The lowest energy obtained is 0.268 μJ per classification. While digital neuromorphic systems achieve high accuracy when training and testing neural networks, their power consumption is still huge in comparison with the biological neural systems.

In contrast to digital neuromorphic systems, analog solutions have saved significant power, thanks to their capability to faithfully mimic biological neural systems. Danneville et al. [12] have proposed an SNN architecture made of 40 analog neurons and 108 plastic synapses. Danneville's system is used to allow the classification of oriented edges. Its silicon core area is 0.025 μm^2 and its overall power consumption is 5 nW. The learning is achieved using the spike time-dependent plasticity (STDP). STDP is a process for which the weight of the synapse between a pre-neuron and a post-neuron is updated based on the relative time between pre-neuron and post-neuron spikes.

To the best of our knowledge, training and testing a network using MNIST in an analog neuromorphic implementation are not addressed in the state-of-the-art. An analog SNN synthesis framework is proposed in Sec. III..

B. Analog Spiking Neural Network: Electronic Neurons

Neuromorphic hardware aims to faithfully mimic biological systems in electronic neurons (illustrated in Fig. 2) and synapses (illustrated in Fig. 3). Electronic eNeurons (eNeurons) are electronic implementations for a mathematical model of a biological neuron. The eNeuron model sets the degrees of complexity and biological accuracy as explained in [11]. Many popular models have been widely used as they offer different trade-offs among energy efficiency (i.e. E_{eff} in J/spike) and computational capabilities. The simplicity of McCulloch-Pitts eNeurons makes them dominant in software implementations, as they only use an activation function to associate an output to the sum of their inputs.

More biologically inspired models mimic the spiking behavior of biological neurons, which are suitable for an analog spiking network implementation. Among them, LIF, and ML models stand out due to their widespread adoption in the literature. LIF eNeurons have a spiking behavior at low complexity by modeling the charge and the discharge of the

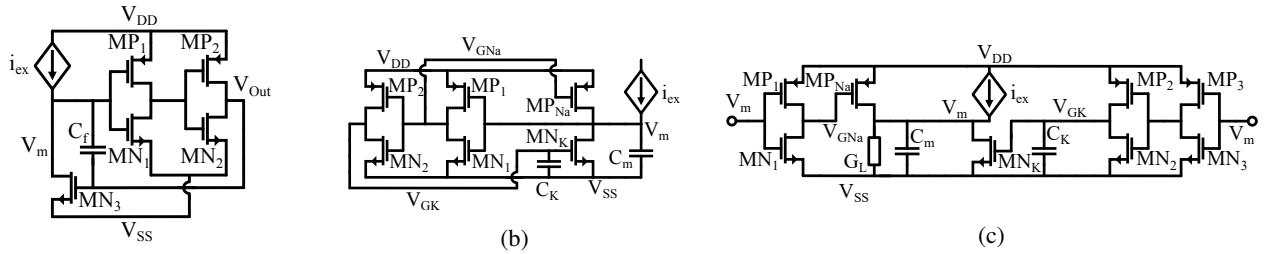


Fig. 2: Schematic illustration of eNeuron circuits studied. (a) LIF model proposed in [7]. (b) simp. ML model proposed in [9]. (c) bio. ML model proposed in [10]

membrane of a neuron. The LIF eNeuron follows a mathematical model described in the literature as

$$V_m = \frac{1}{C_m}\int i_{ex}\left(e^{\frac{V_{DD}-V_m}{\eta\phi_t}} - e^{\frac{-2V_m}{\eta\phi_t}}\right)dt, \qquad (1)$$

where V_m is the spiking voltage signal; i_{ex} is the input excitation current; V_{DD} is the eNeuron supply voltage; η is the subthreshold slope; ϕ_t is the thermal voltage (k_bT/q). A detailed modeling is presented in [22].

Figure 2(a) illustrates a LIF eNeuron proposed by Danneville et al., which is an Axon-Hillock topology [7]. By using parasitic capacitors as the membrane capacitance, Danneville's eNeuron achieved both low energy consumption and low silicon area, which could lead to the design being used in SNNs. Recently, Besrour et al. have also proposed a low power LIF eNeuron in 28 nm CMOS technology in [8]. Besrour's proposal achieved a better area and f_{spike} trade-off in comparison to Danneville's for a total 1.2 fJ/spike energy efficiency.

The ML model provides behavior closer to biological neurons by using multiple complex non-linear differential equations. ML eNeurons have a mathematical model described in the literature, being a non-linear function

$$V_m = -V_{DD}\cdot\tanh\left(\frac{G_L\cdot\sum i_{ex}}{\eta\phi_t} + \frac{1}{2}\ln\left(\frac{G_N}{G_P}\right)\right), \qquad (2)$$

where G_L is the equivalent conductance between eNeuron membrane and ground; G_N/G_P is the transistors' aspect ratio (NMOS conductance over PMOS conductance). A detailed modeling is presented in [9].

ML eNeurons have recently been used to implement artificial neural networks, achieving tasks such as edge recognition [12] and audio signal processing [10]. Two common concerns in such edge computing applications are circuit area and energy efficiency, which is often used as a figure of merit. State-of-the-art eNeurons [10, 12] are most efficient, when they operate at high f_{spike}, and they become increasingly energy-inefficient as f_{spike} lowers.

Sourikopoulos et al. have proposed a simp. ML eNeuron illustrated in Fig. 2(b). By minimizing supply voltage and membrane capacitance, the authors achieved both low power consumption and high f_{spike} [9]. As a step towards SNNs, Ferreira et al. have implemented a neuromorphic analog spiking modulator, which translated a sample input into a spiking output [10]. Figure 2(c) illustrates the bio. ML eNeuron, first proposed in [9] and later improved in [10] in terms of area and f_{spike}. Recently, Takaloo et al. have studied the simp. ML eNeuron and proposed a design optimization in [23]. State-of-the-art ML eNeuron implementations notably make use of subthreshold transistors to reduce area and power consumption [9, 10, 23].

Literature results highlight the wide variety of performance in different ranges of f_{spike}, using different technologies. Thus, the use of the same technology and simulator for eNeurons will prevent the result from being skewed by technology improvements. For this reason, eNeurons depicted in Fig. 2 shall be implemented with in-house tools, see details in [6]. Literature of eNeurons is summarized in Tab. I

TABLE I.: Energy efficiency comparison of state of the art circuits

Ref.	Model	Techn. (nm)	Area (μm^2)	f_{spike} (kHz)	E_{eff} (fJ/spike)
[9]	bio. ML	65	200	1.2	78.3
[10] (idem)	bio. ML	55	98.6	400	1.95
[7]	LIF	65	31	15.6	2.00
[8]	LIF	28	34	343	1.2
[6] (idem)	LIF	55	28.4	130	1.2
[9]	simp. ML	65	35	25	4
[6] (idem)	simp. ML	55	61.7	243	1.8

C. Analog Spiking Neural Network: Electronic Synapses

Electronic synapse is a key component in spiking neural networks, since neuron input (the post-synaptic signal) and output (the pre-synaptic signal) do not have the same dimensions. It is thus necessary to link neurons using synapses, which convert spiking information in i_{syn} post-synaptic signal. The following spiking neurons take the current i_{syn} as an input and deliver an output frequency f_{spike} modulated by the synaptic weight.

Figure 3(a) illustrates the differential pair integrator (DPI) synapse proposed by Bartolozzi and Indiveri in [24]. The DPI synapse generates the post-synaptic current i_{syn} from differential pair unbalance, which is modulated by the bias current available from V_{Pre} pre-synaptic signal and V_W controlling the synaptic weight. By using transistors operating in subthreshold, the proposed differential pair integrator

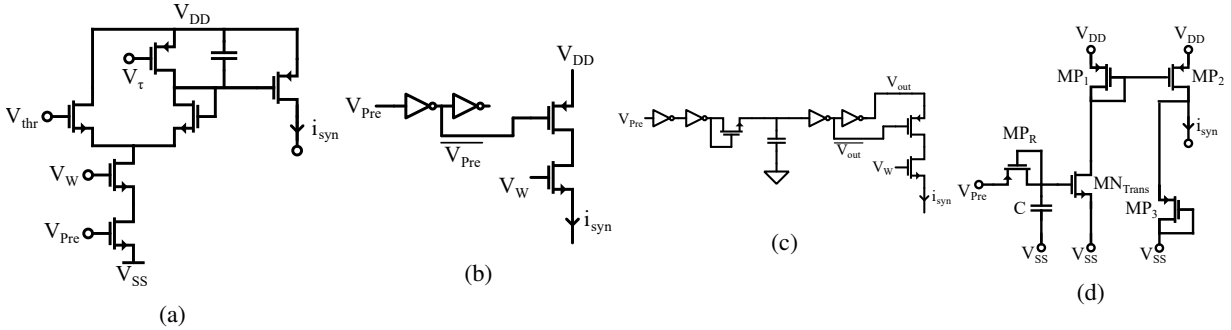


Fig. 3: Schematic illustration of synapse circuits studied. (a) DPI Synapse originally proposed in [24]. (b) Transconductor-based simple synapse. (c) ULP Synapse originally proposed in [12]. (d) Compact Synapse originally proposed in [6].

synapse response is

$$i_{syn}(t) = I_0 \cdot e^{\frac{V_{Pre}(t)-V_{DD}}{\eta \phi_t}}, \tag{3}$$

where I_0 is the leakage current; V_{Pre} is the pre-synaptic signal; i_{syn} is the post-synaptic signal. This model fits closer to the Destexhe mathematical model for synapses [25]. It is remarkable that the mean current output of the synapse is a linear function of the pre-synaptic f_{spike} [24].

In order to build an SNN, Danneville et al. have presented two synapses in [12]. Figure 3(b) illustrates the usage of inverters along with a cascode amplifier to generate the i_{syn}. There, the cascode bias voltage V_W controls the synaptic weight as it modulates the cascode transconductance. Figure 3(c) illustrates a synapse composed of an RC filter with two additional inverters. Those inverters extend the duration of the pre-synaptic pulse to produce a longer post-synaptic current pulse. The transconductance is thus controlled by step voltage pulse generated by the output of the inverters having an amplitude of $2 \cdot V_{DD}$. Danneville's proposal uses PMOS transistors in deep subthreshold, which leads to an ultra-low power consumption as low as the picoWatt [12].

As a step towards deep learning, Azghadi et al. have used memristors to replicate synaptic plasticity [26]. This translated into learning properties and more complex behavior, among which non-linearity of the synapse. However, memristors suffer greatly from noise, an issue amplified by the high density of memristors required for deep learning neural networks. Due to atomic-level random defects and variability in modulation process, memristor characteristics are the main causes of learning accuracy loss in ANNs [27].

The aforementioned synapses have served as an inspiration in previous work [6]. Figure 3(d) illustrates the compact synapse from [6], which is composed of an RC filter, followed by a transconductor and a current mirror. The proposed synapse shall attempt to balance advantages and drawbacks between Destexhe mathematical model agreement, ultra-low power, and plasticity. Plasticity is found in the current mirror architecture from [24], where current-mirror gain is a weighted bias signal. Ultra-low power is observed in the choice of passive RC filter similar as [12]. However, one can observe that eNeurons already have the two inverters as introduced in [12], which can be reused for further power consumption reduction.

Compared to the literature, the compact synapse loses in reconfigurability as it does not rely on a tunable V_W. This loss prevents an online NN learning and force the designer to opt for an offline learning. A NN trained for the MNIST will produce synaptic weights. Such a drawback is largely compensated by the fact that current-mirror gain (i.e., the proposed synapse weight) can be designed to be immune to process variability. Process variability robustness is mandatory in an integrated solution.

Considering a synapse relying on V_W tuning, process variability robustness might only be achieved by a power-consuming analog-to-digital converter controlling (within online learning) the synaptic weight V_W, or by the availability of an external pin. Both solutions are impractical considering the required routing for few hundreds of synapses. For this reason, the authors opt for an offline training using MNIST, following an analog spiking neural inference using the obtained synaptic weights to design the current mirror illustrated in Fig. 3(d).

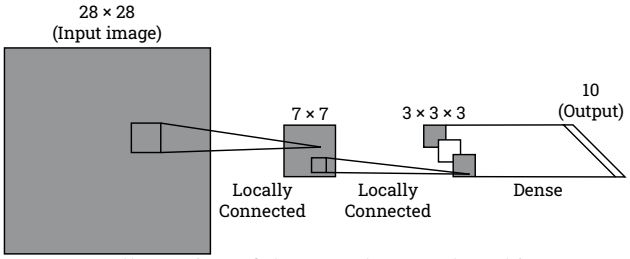


Fig. 4: Illustration of the neural network architecture.

III. Neural Network Synthesis Framework

Aiming at a hardware implementation, electronic circuit constraints must be considered in neural network synthesis. Figure 4 illustrates the proposed neural network synthesis framework. The neural network is trained using 60,000 instances from the MNIST and inferred with 10,000 different instances. Synthesized neural networks shall take as an input layer a 28*28-pixel grayscale picture of a handwritten digit (from 0 to 9) and shall have on its output layer 10 eNeurons, which correspond to the 10 possible digits. As shown in Fig. 4, two hidden layers are located between the input and the output of the neural network.

The first hidden layer is a locally connected layer of 7*7 neurons. It represents the high-level features extracted from the convolution operation applied on the 28*28 image using a kernel. The kernel is basically a filter applied progressively to small regions of the image. For the first layer, the

kernel has a size of 4*4 and has one output filter per neuron. By this convolution operation, a dimension reduction is achieved from a 28*28 image to a 7*7 layer. This allows to greatly reduce the size of the data that has to be processed by the following layers, while retaining useful information. In addition to that, as the neurons are close to their inputs, this choice allows an easier routing.

The second hidden layer is a locally connected layer of 3*3*3 eNeurons. It is obtained using another kernel of 2*2, but with three output filters per neuron. This design is made to extract more useful information, while maintaining a low complexity on the network. Finally, the second hidden layer and the output layer are densely connected. This means that the output layer is a fully connected layer, where all eNeurons from the second hidden layer are connected to the eNeurons of the output layer. Thus, a global inference on the drawn number is made in the output layer, using the local data obtained from the second layer.

To develop this proposal, the mathematical properties of the neural networks shall be considered. Firstly, Subsec. III.A. describes the feasibility studies of the feed-forward neural networks using analog spiking eNeurons. Transfer functions of the eNeuron and synapse are thus presented from software and hardware point of view. Secondly, Subsec. III.B. explores the synthesis framework of the analog spiking neural network based on the activation function and weight analysis. Fitting functions used for the training and the inference in the neural network are then explained.

A. *Feasibility Analysis*

To bridge the gap between software AI and analog neural research, this subsection highlights the tools used for software neural networks and links them with analog spiking neural networks. Deep-learning feasibility in analog spiking NNs is originally studied in previous work [6].

Feed-forward Neural Networks have been a common interest in software-based NN literature. Figure 5 illustrates a common FNN having an input and an output layer. An example network is depicted in Fig. 5(a), displaying a p^{th} hidden layer $(N_{p,i})_{i\leq 4}$. Neurons of a given layer only have the output of the previous layer as an input. The output of a given layer is thus only dependent on the output of the previous layer.

Let it be assumed that a neuron takes an input i and provides an output f. The relation between input and output can be described with an activation function h, which provides the relation $f = h(i)$ for a neuron. Given that in an FNN, neurons are organized in layers (see Fig. 5), one may describe the inputs and outputs of every neuron as vectors for the layer as

$$\begin{aligned} I &= (i_1, i_2, \ldots, i_p)^T \\ F &= (f_1, f_2, \ldots, f_p)^T = H(I), \end{aligned} \tag{4}$$

where capital letters are used for a matrix description of the neural network. Furthermore, as the inputs of the neurons are a linear combination of the outputs of the previous layer, one can define a matrix C_p such that, with F_p as the output of the p^{th} layer, $\forall\ p \in [1,N]$, $F_p = H_p(C_p \cdot F_{p-1})$. Assuming that H_p is linear, it can be expressed as the matrix $H_p = diag(h_{p1}, h_{p2}, ..., h_{pM})$, with M the number of neurons in the layer. F_p can thus be expressed as a matrix product of F_{p-1}, where $F_p = H_p C_p \cdot F_{p-1}$. This leads by induction to the property of FNNs [28]

$$F_N = \prod_{p=1}^{N} H_p C_p \cdot I_1. \tag{5}$$

The output F_N of the FNN is given by a linear combination of the input I_1 for the input layer. It is possible to find an equivalent two layers FNN, as represented in Fig. 5(b). Having a linear activation functions, it thus hinders FNN to have the properties of a deep neural network, even with multiple hidden layers.

While (5) and FNN properties are widely known in the case of mathematical neural networks [2], this is not the case of electronic spiking neural networks (eNN). Figure 6 illustrates a system model for an eNeuron (eN_{pk}) connected to a synapse array $(S_{p,ki})_{i\leq 4}$. eN_{pk} represents the k^{th} eNeuron on the p^{th} layer. As shown in the Fig.6, an eNeuron eN_{pk} takes as an input current i_{ex}, the sum of the current fed ($\sum_{i=1}^{4} i_{(p-1),ik}$) in the node V_m. It outputs a spiking voltage V_m having a *tanh* shape and in which the amplitude of i_{ex} is coded in the f_{spike}, i.e. spiking modulation. A function h can thus be defined for the eNeuron such that $f_{spike} = h(i_{ex})$. Conversely, the synapse circuitry behaves as a frequency-to-current converter, as it converts f_{spike} pre-synaptic information to a post-synaptic current i_{syn}. From the i_{syn} and f_{spike} relationship, a function g can be defined such that $i_{syn} = g(f_{spike})$.

Considering a software-based NN model as Fig. 5(a) and the eNN modeling from Fig. 6, one can study the feasibility of eNN synthesis using literature formalism. The presence of a synapse in eNN linking the neurons modifies the proof in (5). To study the eNN feasibility, the k^{th} eNeuron on the p^{th} layer will be referred to as eN_{pk}, with the transfer function h_{pk}, input i_{pk}, and spiking frequency f_{pk}. The synapse connecting eN_{pk} and $eN_{(p+1)l}$ (the p^{th} eNeuron on the $p+1^{th}$ layer) will be referred to as $S_{p,kl}$, with the transfer function $g_{p,kl}$, input $f_{p,kl}$, output $i_{p,kl}$. Let M be the number of neurons in layer p.

Assuming the transfer functions of the synapses and neurons are linear, i.e. for all p,k,l such that $eN_{pk}, eN_{(p+1)l} \in eNN$, one can find the constants k_{pk}, b_{pk}, $w_{p,kl}$, and $c_{p,kl}$ such that:

$$h_{pk}(i_{ex}) = k_{pk} \cdot i_{ex} + b_{pk} \tag{6}$$

$$g_{p,kl}(f_{spike}) = w_{p,kl} \cdot f_{spike} + c_{p,kl}. \tag{7}$$

The spiking frequency of $eN_{(p+1)l}$ is then

$$f_{(p+1)l} = h_{(p+1)l}\left(\sum_{k=1}^{M} i_{p,kl}\right)$$

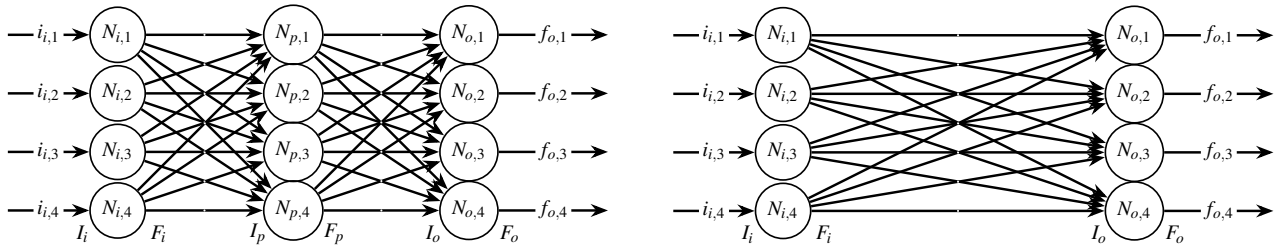
$$f_{(p+1)l} = k_{(p+1)l} \cdot \left(\sum_{k=1}^{M} w_{p,kl} \cdot f_{pk} + c_{p,kl}\right) \tag{8}$$

$$+ b_{(p+1)l} \text{ from (7).} \tag{9}$$

If this relationship is linear, which means that if one expresses the spiking frequencies of a p^{th} layer of eN as a vector, then matrices can be found such that

$$F_p = (f_{p1}, ..., f_{pM})^T \tag{10}$$

$$F_p = A_p F_{p-1}. \tag{11}$$



(a) (b)

Fig. 5: Feed-Forward Neural Network (FNN) illustration, where (a) is a p hidden-layers software NN example and (b) is an equivalent two-layer (input and output) NN. If linear activation functions are available, an NN with no hidden layer such as (b) can be found with the same transfer function as any (a) NN.

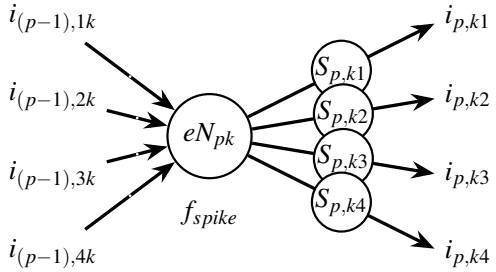


Fig. 6: Analog spiking neuron and synapses models, which can be synthesized using eNeuron and synapse described in Subsec II.B. and Subsec II.C.

By induction, one may obtain

$$I_1 = (i_{11}, ..., i_{1p})^T$$
$$F_N = \prod_{p=1}^{N} A_p \cdot I_1. \tag{12}$$

This proves that the previous result for software NN still applies in the case of an eNN. If published eNeurons and synapses have linear transfer functions, any multi-layer feed-forward eNN is equivalent to a two-layer eNN.

For deep learning purposes, a non-linear activation function is needed. In literature, the non-linear functions *sigmoid*, *ReLU*, and *tanh* are often used as activation functions [2]. The non-linear *sigmoid* function can be related to the eNeuron models (1) and (2) as proposed in the following Subsec. II.B..

B. Synthesis Framework

Feasibility analysis is built on support arguments using software-based neural networks studies. To illustrate the performance trade-offs induced by the previous proof (12), it is necessary to transform the transfer functions of the eNeuron and synapse into an activation function and weight that are usable in synthesis tools like TensorFlow [13]. Nevertheless, modeled activation function must reproduce the FoM observed in the literature [7, 9, 10].

This paper proposes the synthesis framework illustrated in the pseudocode shown in 1. The framework first requires a `structure` construction for the neural network as illustrated in Fig. 4. One may choose a bias to be implemented at the neuron's input to assess the real-world impact of the eNN feasibility proof depicted in the previous subsection. First the bias for each neuron is fixed at 0 to limit the complexity of the network ($x_{min} = 0$, no restriction). Later, to enhance the energy efficiency (E_{eff}), a bias of 0.2 a.u. is preferred ($x_{min} = 0.2$, restricted area). A general solution might need to add a current mirror to the proposed synapse having both excitation and inhibition effects in neurons. However, in this proposed technique, the weights are kept positive to achieve only an excitation purpose as presented in Fig. 3(d). The reason behind that is to avoid the mismatch of the transfer functions of the inhibition and excitation synapses which could severely impact accuracy of the hardware network.

The combined transfer function of the neuron and synapse can be written as $i_p = h(g(i_{p-1}))$, where i_p and i_{p-1} are the excitation currents in F_p and F_{p-1} hidden layers extracted from PLS simulations of eNeuron and synapse models (see Fig. 6 and the pseudocode 1). Assuming an ideal current mirror, the function h can be written as the product of a non-linear function and a constant current gain, leading to:

$$i_p = h_n(g(i_{p-1})) \cdot \frac{(W/L)_1}{(W/L)_2}, \tag{13}$$

where $\frac{(W/L)_1}{(W/L)_2}$ is the current mirror gain obtained from MP_1 and MP_2 transistors depicted in Fig. 3(d). One may conclude that it is quite a realistic activation function as (13) can be derived from post-layout simulations in [10] or even measurements [7, 9]. Nevertheless, this physical modeling is not suitable for eNN synthesis considering the procedure described in Subsec. III.A..

The proposed synthesis framework learns on a normalized activation function to improve training from literature tools [13]. Thus, the previous equation can be rewritten as:

$$y = \left(\frac{h_n(g(x))}{max_x(h_n(g(x)))}\right) \cdot \left(\frac{(W/L)_1}{(W/L)_2} \cdot max_x(h_n(g(x)))\right). \tag{14}$$

where y and x are the normalized excitation currents i_p and i_{p-1}. The activation function f and the weights w can thus be linked to the transfer functions and W/L ratios as

$$f(x) = \frac{h_n(g(x))}{max_x(h_n(g(x)))} \text{and} \tag{15}$$

$$w = \frac{(W/L)_1}{(W/L)_2} \cdot max_x(h_n(g(x))). \tag{16}$$

The relation (15) gives the activation function which is implemented in the training framework. The relation (16) allows for extracting the $(W/L)_i$ ratios from the weights

Algorithm 1 eNN Synthesis Framework

```
Structure = [
locally_connected7*7(filters=1,kernel=(4,4), strides=(4,4))
locally_connected3*3*3(filters=3,kernel=(2,2), strides=(2,2))
dense(outputs=10)]
model_restriction (x_min, 0 || 0.2)
model = load_results (PLS_eNeuron, PLS_synapse)
[i_p , i_p−1] = model_eNeuron*model_synapse
[y,x] = normalized ([i_p , i_p−1])
polynomial_fit = polyFit ([i_p ,i_p−1], R^2 ≈ 1.0, order=12)
sigmoid_fit = sigmoidFit ([i_p , i_p−1] , R^2 ≈ 0.9)
model_training=(structure,sigmoid_fit, model_restriction)
for epoch = 1 to 100 do
    tensor_weight = TensorFlow_learning (model_training, MNIST, epoch)
    accuracy_training = model_inference (tensor_weight, model_training)
    if accuracy_training ≥ 0.9 then
        trained_weight = tensor_weight
    else if epoch==100 then
        trained_weight = tensor_weight
    end if
end for
w̄ =trained_weight
σ_w = 0.01*trained_weight
statistic_model = normal_distribution (w̄t, σ_w)
accuracy_inference = MonteCarlo_inference (statistic_model, structure, polynomial_fit, model_restriction)
```

w_i in the software neural network. However, there is no access for such activation function (15) in TensorFlow. Thus, two fits are made for each neuron, to create functions usable for training and inference in software neural networks. As described in the Algo. 1, the polynomial fitting model (`polynomial_fit`) is used for inference (`accuracy_inference`) after the eNN is already trained (`model_training`) with sigmoid fit (`sigmoid_fit`, second fitting).

The first fitting is a polynomial fit. This allows a very close fit in the region where the function varies, while being unvarying in the constant regions. This function must be a very close fitting ($r^2 \approx 1.0$). However in this case, this polynomial fit leads to issues during training due to the brutal change of slope near 0.

The second fit is a generalized logistic function first introduced in [29]. The generalized logistic function or curve is an extension of the logistic or sigmoid functions, from this point it shall be named sigmoid fit. The fitting function used is

$$y(x) = A + \frac{K-A}{(C+Qe^{-Bx})^{1/\nu}}, \tag{17}$$

where A, B, K, C, Q, ν are fitting constants. While the sigmoid fit is less precise, it provides a function with a more moderate slope. Thus, (17) is suggested to provide a close enough fit ($r^2 \approx 0.9$), while still allowing the use of gradient descent training algorithms. The reason to that is that sigmoid fit is C^∞ everywhere which improves the tool's convergence.

A realistic activation needs a precise estimation of the activation function f according to (15), and also to consider the synaptic weight formulated in (16) as realistic current mirror implementation (i.e. current mirror mismatch). To this end, the inference shall include the process variability of current mirror gain in (16). `Monte Carlo inference` simulates trained eNN under process variability (`statistic_model`). To this end, it is proposed to turn the synaptic weights into statistic variables following a normal distribution of $\bar{w} =$ `trained_weight` from training phase and a standard deviation of $\sigma_w = 0.01 \cdot$ `trained_weight`. This allows for assessing the consequences of transistor variability considering current-mirror mismatch while implementing the weights.

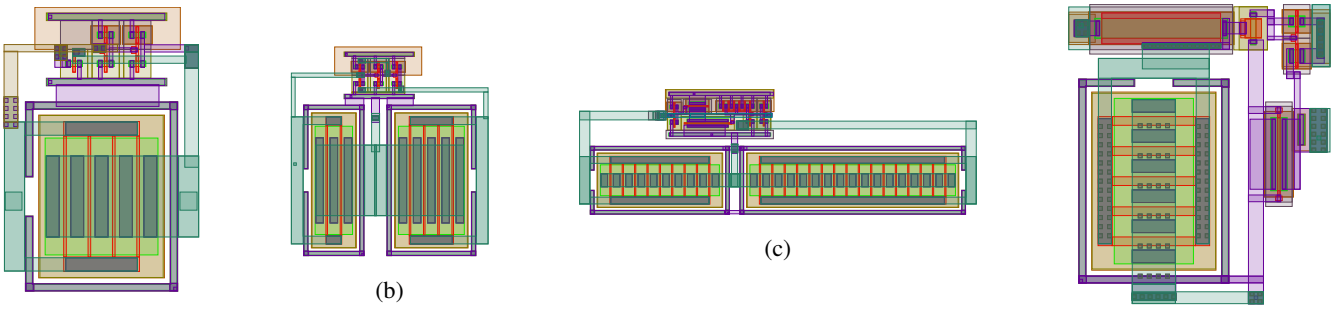
IV. Synthesis Illustration

A. Transistor-level eNeuron and Synapse Design

Using the BiCMOS 55 nm technology, a transistor-level design is proposed for eNeurons in Fig. 2. Analog spiking eNN synthesis shall be considered for a total of three synthesis illustrations being: LIF, simplified ML (simp. ML), and biomimetic ML (bio. ML). Table II summarizes the eNeuron and compact synapse transistors' sizing. Figure 7 illustrates the proposed layouts for eNeurons and compact synapse. Found area consumption is as follows: LIF has $6.56 \times 4.33\ \mu m^2$ in Fig. 7(a); simp. ML has $8.21 \times 7.52\ \mu m^2$ in Fig. 7(b); bio. ML has $5.69 \times 17.33\ \mu m^2$ in Fig. 7(c); synapse has $6.60 \times 7.55\ \mu m^2$ in Fig. 7(d). Depicted layout is presented in a high-quality vectorial image using [30], which improves readability and reproducibility of presented results.

TABLE II.: Sizing of MN and MP transistors in $W \times L$ (nm) and in capacitance for C (fF) for the neurons

LIF model			
MP_1, MN_1, MP_2, MN_2, MN_3			135×65
C_f	5.0		
simp. ML model			
MP_1, MN_1, MP_2, MN_2, MP_{Na}, MN_K			135×65
C_m	4.0	C_K	8.0
bio. ML model			
MP_1	135×60	MN_1	200×60
MP_2	1200×60	MN_2	135×60
MP_3	200×60	MN_3	135×60
MP_{Na}	800×60	MN_K	2000×60
C_m	5.5	C_f	9.8
MP_d	$500 \times 10,000$	$MN_{K'}$	2000×60
C'_K	6.8	G_L	Leakage
Compact synapse			
MP_R	500×3000	MN_{Trans}	135×480
MP_1	270×60	MP_2	324×60
C	11.9	MP_3	1840×60



(a) (b) (c) (d)

Fig. 7: In-house proposed layouts for eNeurons under test: (a) LIF model having $6.56 \times 4.33\ \mu m^2$, (b) simp. ML model having $8.21 \times 7.52\ \mu m^2$, (c) bio. ML model having $5.69 \times 17.33\ \mu m^2$. Compact synapse layout (d) having $6.60 \times 7.55\ \mu m^2$.

B. *Post Layout Simulation Results*

PLS are carried out using Spectre Virtuoso from BSIM4 models, which come from measured-based foundry models [31]. An i_{ex} parametric sweep is PLS simulated between 1 pA and 1 nA to extract the f_{spike} response of the eNeurons and its root-mean-square power consumption (P_{RMS}). At $i_{ex} > 1\ nA$, f_{spike} begins to fall due to circuit stability due to subthreshold biasing (i.e. spiking oscillations stops). Thus, the pointed i_{ex} range corresponds to the estimated dynamic of the eNeurons. Energy efficiency is calculated as $E_{eff} = P_{RMS}/f_{spike}$. For PLS results, the synapse is used to connect two eNeurons, thus assuring the correct operation of the synapse. Figure 8 illustrates the obtained results.

B..1 eNeuron PLS results show that the relationship between i_{ex} and f_{spike} is non-linear, with E_{eff} significantly improving on increased input current. As E_{eff} is an important concern when designing and using eNeurons, one may expect the eNeurons to be used in operating areas of high E_{eff}. Thus, a linear fit was realized for those areas in particular, omitting the less efficient low-spiking frequencies. Having a smaller range of operation, the LIF eNeuron is significantly more non-linear than the other eNeurons. No satisfying linear fit could be found for the transfer function in Fig. 8(a), for any significant i_{ex} range.

For the eNeuron designed from simp. and bio. ML models, a satisfying linear fit was found ($r^2 = 0.99$) for $i_{ex} > 200$ pA. While the transfer functions are still non-linear in this area, they can be closely approximated by a linear function. The difference between the linear fits and real transfer functions on average of 1.58% of spiking frequency for simp. ML in Fig. 8(b), with the fit $f(i) = 0.10 \times i + 158.5$, and 3.09% for bio. ML in Fig. 8(c), with the fit $f(i) = 0.28 \times i - 14.0$.

Results also show the E_{eff} reduction over eNeuron i_{ex} sweep. Figure 8(b) shows that the eNeuron becomes most energy efficient above 100 pA, with E_{eff} strongly degrading for small i_{ex}. Figure 8(c) shows that the eNeuron becomes energy efficient at $i_{ex} > 200$ pA. The eNeurons are expected to be used in ranges above 100 pA and 200 pA respectively.

Aforementioned results highlight a trade-off between energy efficiency and deep-learning capabilities while using state-of-the-art eNeurons. If a designer chooses to use the full range frequency of the eNeurons, he will face poor E_{eff}. Instead, if one chooses to limit the operating range to higher f_{spike}, then the eNeurons will have a linear characteristic. In this case, he will either be limited to the processing capabilities of a shallow neural network, or have to use a non-linear synapse. A designer might use the LIF eNeurons presented in [7], which provide non-linearity at the expense of dynamic range. This reasoning led the proposed technique to be implemented either in $x_{min} = 0$ (no restriction) or $x_{min} = 0.2$ (restricted area).

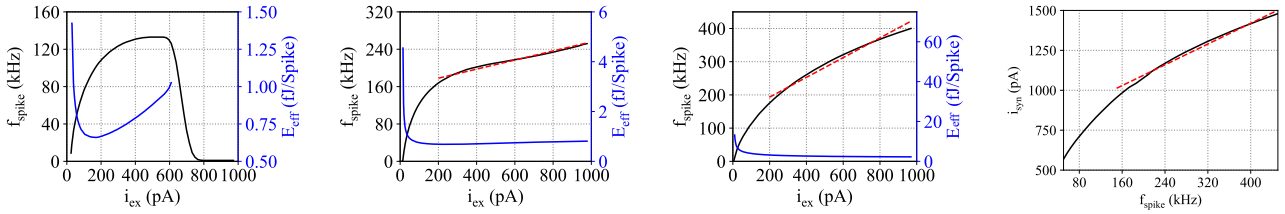
B..2 Compact synapse PLS results exhibit the necessary functions: output current grows with input spiking frequency, and spiking injection loops are prevented. The excitation current of the input neuron was swept from 0 to 1 nA, to generate a sweep of the input spiking frequency of the synapse. Figure 8(d) shows that the output current of the synapse increases with the input frequency, which is consistent with the expected behavior for an excitation synapse. The frequency range explored is 0 to 450 kHz, which is the output range of the bio. ML eNeuron, i.e. the larger dynamic range. While the output is non-linear, a linear fit can be found for a specific input range. For $f_{spike} > 160$ kHz , the linear fit $i(f) = 1.63 \times f + 769.2$ was found with $r^2 > 0.99$, with an average difference between linearization and output of 1.09%.

As a linear fit can be found for the synapse in areas of high energy efficiency of the eNeurons, it will not introduce non-linearity for networks functioning in those operating points. This means that the transfer function between the outputs of two layers will be a linear combination. This makes deep learning and energy efficiency mutually exclusive if those ultra-low power neuromorphic devices are used.

C. *Activation Function Results*

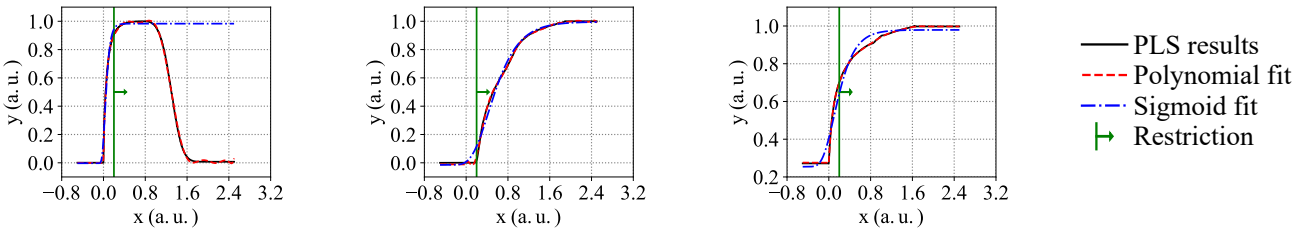
Presented PLS are complete but complex for a neural network training. A monotone increasing of the post-synaptic signal as a function of the pre-synaptic signal is required as presented in [32]. Figure 9 shows the activation function derived from the combination of the synapse and the different neurons, described in the framework as `normalized` function using arbitrary units (a.u.). It is considered the excitation current i_{p-1} in the F_p hidden layer as a pre-synaptic signal in x-axis and the excitation current i_p in the F_{p-1} hidden layer as a post-synaptic signal in y-axis. Represented in black continuous line, normalized PLS results are presented for LIF (see Fig. 9(a)), simp. ML(see Fig. 9(b)), and bio. ML (see Fig. 9(c)) eNeurons.

As expected, one may notice that activation function are highly non-linear. The first fit is a polynomial fit of degree



(a) (b) (c) (d)

Fig. 8: Post-Layout simulation results using BiCMOS 55nm technology. Black continuous line illustrates the f_{spike} in kHz over i_{ex} variation between 1 pA and 1 nA. Blue continuous line illustrates the E_{eff} calculated from P_{RMS}/f_{spike} in fJ/spike. Red dashed line represents a linear fitting at high f_{spike} and low E_{eff} with $r^2 = 0.99$. (a) LIF eNeuron FoM, where no linear fitting is found. (b) Simp. ML eNeuron FoM, were a linear fit is obtained for $i_{ex} \geq 200$ pA. (c) Bio. ML eNeuron FoM, were a linear fit is obtained for $i_{ex} \geq 200$ pA. (d) Compact synapse FoM, where a linear fit is obtained for $f_{spike} \geq 160$ kHz.



(a) (b) (c)

Fig. 9: Polynomial and sigmoid fit for the activation function of the studied synapse and neurons using arbitrary units (a.u.). Black continuous line presents the PLS results. Blue dashed line represents the fitted sigmoid activation function. Red dashed line represents a 12^{th} order polynomial fitting, which is considered in this work as an realistic activation function. Green x-axis bar highlights the dynamic range restriction where low E_{eff} is found. Results are obtained for different eNeuron models: (a) LIF, (b) simp. ML, (c) bio. ML.

12^{th}. This allows for a very close fit in the region where the function varies, while being constant in the constant regions. This very close fit ($r^2 \geq 0.999$) leads to issues during training due to the brutal change of slope near 0. However, it must be used for inference while the eNN is already trained with the sigmoid fit. Thus results will be closer to the behavior of the hardware-based SNN using published eNeurons [6]. Red dashed lines represent the obtained polynomial fit for the following LIF (see Fig. 9(a)), simp. ML(see Fig. 9(b)), and bio. ML (see Fig. 9(c)) eNeurons.

The second fit using the generalized logistic function aims fit less precise but it provides a function with a more moderate slope. Blue dash-dotted lines represent the obtained sigmoid fit for the following LIF (see Fig. 9(a)), simp. ML (see Fig. 9(b)), and bio. ML (see Fig. 9(c)) eNeurons. LIF sigmoid fit achieves a $r^2 \geq 0.158$, due to the disagreement found for $x \geq 0.8$ a.u. limiting LIF dynamic range. Sigmoid fit for simp. and bio. ML models achieve $r^2 \geq 0.995$ and $r^2 \geq 0.985$ respectively.

A designer, interested in minimizing E_{eff}, may be driven to seek an $i_{ex} \geq 200$ pA (see Fig. 8). For this end, normalized activation functions shall be considered for $x \geq 0.2$ a.u. for polynomial and sigmoid fitting. Restricting the input of the functions leads to a lower degree of non-linearity. A green vertical bar highlights the low-power restriction imposed on the activation functions for LIF (see Fig. 9(a)), simp. ML(see Fig. 9(b)), and bio. ML (see Fig. 9(c)) eNeurons.

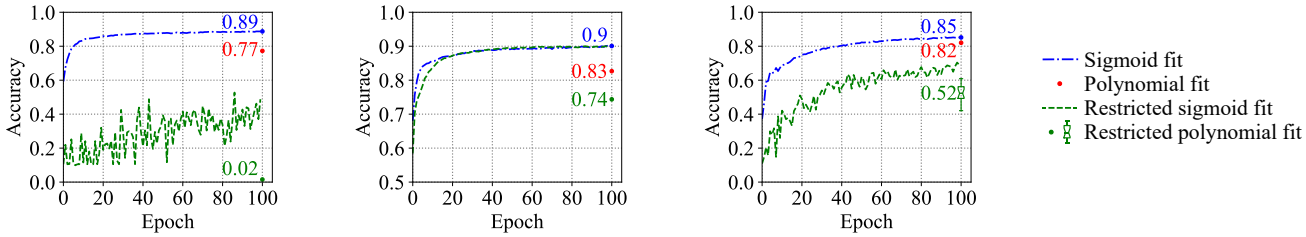
D. eNN Training

For each eNeuron, the neural network is trained for 100 epoch with the sigmoid activation function. The ideal sigmoid response enables the use of TensorFlow and build a machine learning application much faster for the MNIST. Figure 10 illustrates the sigmoid learning in blue dashed-dot line. Nevertheless, eNeuron response studied in Subsec. III.C. is not an ideal sigmoid.

The sigmoid-trained architecture and synaptic weights are then simulated using the polynomial activation function depicted in Fig. 9. Thus, eNN accuracy can be estimated using more realistic components from Monte Carlo simulations. Red dot markers in Fig. 10 illustrate the inference phase of the trained eNN using the `statistic_model`.This result assess the consequences of transistor variability considering current-mirror mismatch while implementing the weights.

Moreover, a designer can be motivated for strong reduction of eNN energy efficiency and may choose to train the neural network in the restricted area green-highlighted in Fig. 9 where the lowest E_{eff} is found. Restricted sigmoid training is presented in dashed green line, while Monte Carlo results considering the polynomial activation function are shown using a green marker in Fig. 10. A green error bar is preferred when Monte Carlo results reveals an accuracy distribution bigger than the mathematical resolution (i.e. two digits in this work).

Figure 10(a) highlights a significant accuracy improvement until epoch 20 which slows down afterward for the LIF eNeuron model. This results highlight that the LIF eNeuron



(a) (b) (c)

Fig. 10: Accuracy asserted for neural network trained for 100 epoch. Blue continuous line presents the learning procedure using sigmoid activation function. The red dot is the estimated accuracy when activation function is replaced by the polynomial fitting obtained from PLS results. Blue dashed line represents the learning procedure using an restricted area of the sigmoid activation function. Blue marker is the estimated accuracy when activation function is replaced by the polynomial fitting while considering process variation. Results are obtained for different eNeuron models: (a) LIF, (b) simp. ML, (c) bio. ML.

has the fastest training convergence compared to the other two eNeurons under test. The Monte Carlo simulation using the polynomial activation function (realistic activation function) shows a decrease of 12% of the mean accuracy ($3 \cdot \sigma \leq 0.01$) exhibiting a significant decrease in performance for an integrated eNN solution. Restricted training above 0.2 a.u. excitation is capable of improving average E_{eff}. The performance of such a low-power eNN with the sigmoid fit improves erratically until epoch 100. This lack of convergence, accuracy lower than 50%, is probably due to the limited dynamic range of LIF eNeurons. Moreover, using realistic activation function only highlights a performance drop below randomness turning the eNN useless.

Figure 10(b) illustrates a significant accuracy improvement until epoch 40 and reaching 0.9 at epoch 100 for simp. ML eNeuron model. Besides, the realistic activation function simulation reveals a means accuracy decrease of 7% (ie. $\mu = 0.83$, $3 \cdot \sigma \leq 0.01$) . This is the best accuracy found in this work. Restricted training above 0.2 a.u. excitation still leads to training convergence to the accuracy of 0.9. However, the realistic activation function simulation highlights a loss of accuracy of 16%, making the network significantly less accurate.

Figure 10(c) illustrates a significant accuracy improvement until epoch 40, then continues to improve slowly for bio. ML eNeuron model. The difference of performance between the ideal sigmoid and the realistic activation function simulation is only 3%, making the loss of accuracy minimal. It is found that in the case of bio. ML eNeuron model, the training using the sigmoid fit is more transferable to the polynomial fit. The weights are thus closer to the realistic ones than in the case of LIF eNeuron. Restricted training above 0.2 a.u. excitation improves more erratically up to 0.7 of accuracy. However, the realistic activation function simulation highlights an accuracy loss to a $\mu = 0.52$ and a $3 \cdot \sigma = 0.12$.

E. eNN Comparison and Discussion

This analysis concludes that LIF eNeuron implementation, due to the limited biological inspiration, presents an important dynamic range limitation. Considering the polynomial activation function depicted in Fig. 9, simp. and bio. ML eNeurons achieve accuracy approximately of 0.82. Moreover, Monte Carlo simulations also show that a mismatch of 1% does not have a significant effect on eNN accuracy.

The mutually exclusive trade-off between the deep learning and ultra-low power is illustrated in the previous subsection by imposing a model restriction ($x_{min} = 0.2$). Thus, a novel performance trade-off is revealed when limiting the dynamic range of the neuron to reduce E_{eff}. To achieve a trained eNN with a low E_{eff}, one must use a proper neural network model considering the non-linear characteristics of the activation function.

Table III highlights the trade-off between the area and the total power consumption while considering active 86 eNeurons and 1238 synapses. Total power is obtained from active eNeuron PLS. The idle power consumption is calculated for the condition no input in I_i layer (i.e. leakage current only). Area is obtained from eNeuron and compact synapse area (see Fig. 7 for details). Besides, presented estimation also considers the area overhead on current mirror implementation, which depends on synapse weight.

The LIF eNeuron implementation achieved the lowest area and power consumption. Both simpl. and bio. ML eNeuron implementation achieved a total power twice bigger than the LIF eNeuron one. The idle power consumption is around 140 nW among three proposed syntheses, as it is dominated by the synapses idle power consumption. A similar conclusion was previously found in [12], in which idle power consumption per synapse were 30 pW while here it is 112 pW with a much larger number of synthesized synapses.

V. Conclusions

To bridge the gap between software AI and analog neural network, a framework with an algorithm guideline for energy efficient analog SNN synthesis is proposed. It uses machine learning TensorFlow to train and evaluate the accuracy of an analog SNN composed of 86 eNeurons and 1238 synapses with two hidden layers. MNIST is considered to achieve a general solution for SNN implementation on analog hardware. Results show that an SNN using ML eNeurons is more accurate (0.82) than the one using LIF eNeurons (0.77) but at the cost of higher-power consumption. Furthermore, results from Monte Carlo simulations show that a mismatch of 1% imposed on the synaptic weights, which are the critical factor of training the neural network, does not significantly affect the accuracy.

TABLE III.: Estimated surface and power consumption for Synthesized eNN

	LIF		simp. ML		bio. ML	
	Area (μm^2)	Total [idle] (nW)	Area (μm^2)	Total [idle] (nW)	Area (μm^2)	Total [idle] (nW)
86 eNeurons	2458	4.10 [1.36]	5313	11.63 [1.38]	8489	77.10 [7.96]
1238 Synapses	3291	251.8 [139.4]	3826	313.3 [139.4]	3640	510.0 [139.4]
Total	5749	255.9 [140.7]	9139	522.43 [140.8]	12130	587.9 [147.4]

ACKNOWLEDGMENT

The authors would like to thank Joao Frischenbruder Sulzbach, who was co-author in the SBCCI paper, which is extended in this work. He provided layout material presented here.

REFERENCES

[1] A. Shrestha, et al., "A Survey on Neuromorphic Computing: Models and Hardware," *IEEE Circuits and Systems Magazine*, vol. 22, no. 2, pp. 6–35, 2022. [Online]. Available: https://doi.org/10.1109/MCAS.2022.3166331

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, dec 1989. [Online]. Available: http://link.springer.com/10.1007/BF02551274

[3] J. Park, et al., "Hierarchical Address Event Routing for Reconfigurable Large-Scale Neuromorphic Systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2408–2422, oct 2017. [Online]. Available: https://doi.org/10.1109/TNNLS.2016.2572164

[4] E. E. Tsur and M. Rivlin-Etzion, "Neuromorphic implementation of motion detection using oscillation interference," *Neurocomputing*, vol. 374, pp. 54–63, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2019.09.072

[5] S. Moradi, et al., "A Scalable Multicore Architecture with Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb 2018. [Online]. Available: https://doi.org/10.1109/TBCAS.2017.2759700

[6] T. Soupizet, et al., "Deep Neural Network Feasibility Using Analog Spiking Neurons," in *Proc ACM IEEE Symp. Integr. Circuits Syst. Design*, Porto Alegre, RS, Brazil, Aug 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/SBCCI55532.2022.9893216

[7] F. Danneville, et al., "A Sub-35 pW Axon-Hillock artificial neuron circuit," *Solid-State Electron.*, vol. 153, no. 1, pp. 88–92, 2019. [Online]. Available: https://doi.org/10.1016/j.sse.2019.01.002

[8] M. Besrour, et al., "Analog Spiking Neuron in 28 nm CMOS," in *IEEE New Circuits Syst. Conf.*, Quebec, Canada, Jun 2022. [Online]. Available: https://doi.org/10.1109/NEWCAS52662.2022.9842088

[9] I. Sourikopoulos, et al., "A 4-fJ/spike artificial neuron in 65 nm CMOS technology," *Frontiers in Neuroscience*, vol. 11, no. 123, pp. 1–14, mar 2017. [Online]. Available: https://doi.org/10.3389/fnins.2017.00123

[10] P. M. Ferreira, et al., "Neuromorphic analog spiking-modulator for audio signal processing," *Analog Integrated Circuits and Signal Processing*, vol. 106, no. 1, pp. 261–276, Jan 2021. [Online]. Available: https://link.springer.com/10.1007/s10470-020-01729-3

[11] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," pp. 1–88, 2017. [Online]. Available: http://arxiv.org/abs/1705.06963

[12] F. Danneville, et al., "Sub-0.3V CMOS neuromorphic technology and its potential application," in *2021 International Conference on Content-Based Multimedia Indexing (CBMI)*, no. 1. Lille, France: IEEE, jun 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CBMI50038.2021.9461899

[13] M. Abadi, et al., "A machine learning approach to radar sea clutter suppression," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*. USENIX Association, nov 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[14] T. Chilimbi, et al., "Project ADAM: Building an efficient and scalable deep learning training system," in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2014*. Broomfield, CO, USA: USENIX, oct 2014, pp. 571–582. [Online]. Available: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/chilimbi

[15] A. Baldominos, Y. Saez, and P. Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, p. 3169, aug 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3169

[16] Y. LeCun, et al., "Learning algorithms for classification: a comparison on handwritten digit recognition," in *Neural networks: the statistical mechanics perspective*, J. Oh, S. Cho, and C. Kwon, Eds., feb 1995, vol. 1, pp. 261–276. [Online]. Available: https://doi.org/10.1142/2808

[17] L. Wan, et al., "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," in *Proceedings of the 30th International Conference on Machine Learning*, D. Dasgupta, Sanjoy and McAllester, Ed., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, Jun 2013, pp. 1058–1066. [Online]. Available: https://doi.org/10.1109/TPAMI.2017.2703082

[18] P. Verbancsics and J. Harguess, "Image Classification Using Generative Neuro Evolution for Deep Learning," in *2015 IEEE Winter Conference on Applications of Computer Vision*. Waikoloa, HI, USA: IEEE, Jan 2015, pp. 488–493. [Online]. Available: https://doi.org/10.1109/WACV.2015.71

[19] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary convolutional neural networks: An application to handwriting recognition," *Neurocomputing*, vol. 283, pp. 38–52, Mar 2018. [Online]. Available: https://doi.org/10.1016/j.neucom.2017.12.049

[20] A. Renner, et al., "The Backpropagation Algorithm Implemented on Spiking Neuromorphic Hardware," Los Alamos National Laboratory, Tech. Rep., 2021. [Online]. Available: http://arxiv.org/abs/2106.07030

[21] S. K. Esser, et al., "TrueNorth(a/b) in YOSO," *Advances in Neural Information Processing Systems*, vol. 2015-Jan, pp. 1117–1125, 2015.

[22] M. Daliri, et al., "A comparative study between E - neurons mathematical model and circuit model," *IET Circuits Devices Syst.*, vol. 15, pp. 175–192, feb 2021. [Online]. Available: https://doi.org/10.1049/cds2.12017

[23] H. Takaloo, A. Ahmadi, and M. Ahmadi, "Design and Analysis of the Morris-Lecar Spiking Neuron in Efficient Analog Implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. pp, no. pp, pp. 10–14, 2022. [Online]. Available: https://doi.org/10.1109/TCSII.2022.3203929

[24] C. Bartolozzi and G. Indiveri, "Synaptic Dynamics in Analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, Oct 2007. [Online]. Available: https://doi.org/10.1162/neco.2007.19.10.2581

[25] A. Destexhe, Z. F. Mainen, and T. J. Sejnowski, "Kinetic Models of Synaptic Transmission," in *Methods in Neuronal Modeling*, 2nd ed., C. Koch and I. Segev, Eds. MIT Press, Cambridge, MA, 1998, pp. 1–26.

[26] M. R. Azghadi, et al., "A Hybrid CMOS-Memristor Neuromorphic Synapse," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 434–445, 2017. [Online]. Available: https://doi.org/10.1109/TBCAS.2016.2618351

[27] W. Xu, J. Wang, and X. Yan, "Advances in Memristor-Based Neural Networks," *Frontiers in Nanotechnology*, vol. 3, no. March, pp. 1–14, 2021. [Online]. Available: https://doi.org/10.3389/fnano.2021.645995

[28] K. Guo, et al., "Software-Hardware Codesign for Efficient Neural Network Acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, Mar 2017. [Online]. Available: https://doi.org/10.1109/MM.2017.39

[29] F. J. Richards, "A Flexible Growth Function for Empirical Use," *Journal of Experimental Botany*, vol. 10, no. 2, pp. 290–301, 1959.

[30] J. R. R. O. Martins, F. Alves, and P. M. Ferreira, "IC-Layout Render : Image rendering tool for integrated circuit layout in Python," *Open Software*, vol. pp, no. pp, pp. 1–5, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5618268

[31] P. Chevalier, et al., "A 55 nm triple gate oxide 9 metal layers SiGe BiCMOS technology featuring 320 GHz f_T / 370 GHz f_{MAX} HBT and high-Q millimeter-wave passives," in *Int. Electron Devices Meeting, IEDM*, San Francisco, CA, USA, dec 2014, pp. 3.9.1–3.9.3. [Online]. Available: https://doi.org/10.1109/IEDM.2014.7046978

[32] S. Wang, P. M. Ferreira, and A. Benlarbi-Delai, "Behavioral Modeling of Nonlinear Power Amplifiers Using Spiking Neural Networks," in *IEEE New Circuits Syst. Conf.*, Quebec, Canada, jun 2022, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NEWCAS52662.2022.9842167