# Final Report - Efficient On-Device Learning: Hardware-Software Co-Design of a Digital Neural Accelerator Utilizing Local Learning Rules

**Marco P. E. Apolinario**[1], **Kaushik Roy**[1], **Charlotte Frenkel**[2]
[1]*School of Electrical and Computer Engineering, Purdue University*
[2]*Microelectronics Department, Delft University of Technology*

## 1 Summary

Currently, deep neural networks (DNNs) are trained using stochastic gradient descent (SGD) methods in conjunction with the backpropagation (BP) algorithm. BP computes gradients of the loss function with respect to each weight in the model, in order to assign credit or blame (temporally and spatially) to specific synaptic weights. Despite its effectiveness, BP comes with significant computational and memory overheads. For a model with $L$ layers and $n$ neurons per layer, BP has time and memory complexities of $O(Ln^2)$ and $O(Ln)$, respectively. These complexities increase further for models with recurrent connections to $O(TLn^2)$ and $O(TLn)$, where $T$ is the length of the input sequence. This high computational cost makes BP impractical for on-device learning on low-power edge devices. Edge devices, such as IoT devices, require energy-efficient algorithms due to their limited computational resources and battery life. Existing studies (Zhang et al., 2018; Ankit et al., 2020; Peng et al., 2021) emphasize the need for alternative training methodologies that reduce energy consumption by minimizing memory accesses and computational demands. To address these challenges, there is a growing interest in hardware-software co-design approaches that optimize learning algorithms for low-power devices.

**Objectives:** To tackle the limitations of existing methods, this project proposes the development of a digital neural accelerator supporting on-device learning using fully local learning rules. Specifically, we aim to co-design hardware and learning algorithms that integrate temporal locality and spatial locality. The main goals are:

- Design and simulate a low-power neural network accelerator capable of on-device learning using fully local learning rules;
- Evaluate the performance and robustness of these rules, particularly against quantization effects, when applied to deep artificial neural networks on low-power devices.

## 2 Results

This section presents the key outcomes of the project, organized according to the three main phases outlined in the project timeline (Fig. 1): (1) the proposed algorithmic contribution (Phase 1), (2) the hardware-algorithm co-design analysis (Phase 2), and (3) a hardware architecture overview detailing how the learning method maps to digital hardware, based on preliminary results from the ongoing Phase 3.

### 2.1 A Scalable Temporally and Spatially Local Learning Rule for SNNs

The first phase of the project focused on algorithm development. Drawing inspiration from biological principles, such as eligibility traces, spike-timing-dependent plasticity (STDP), and neural synchronization, we introduce TESS, a learning rule that is both temporally and spatially local, designed for training Spiking Neural Networks (SNNs). TESS achieves both temporal and spatial credit assignment by relying exclusively

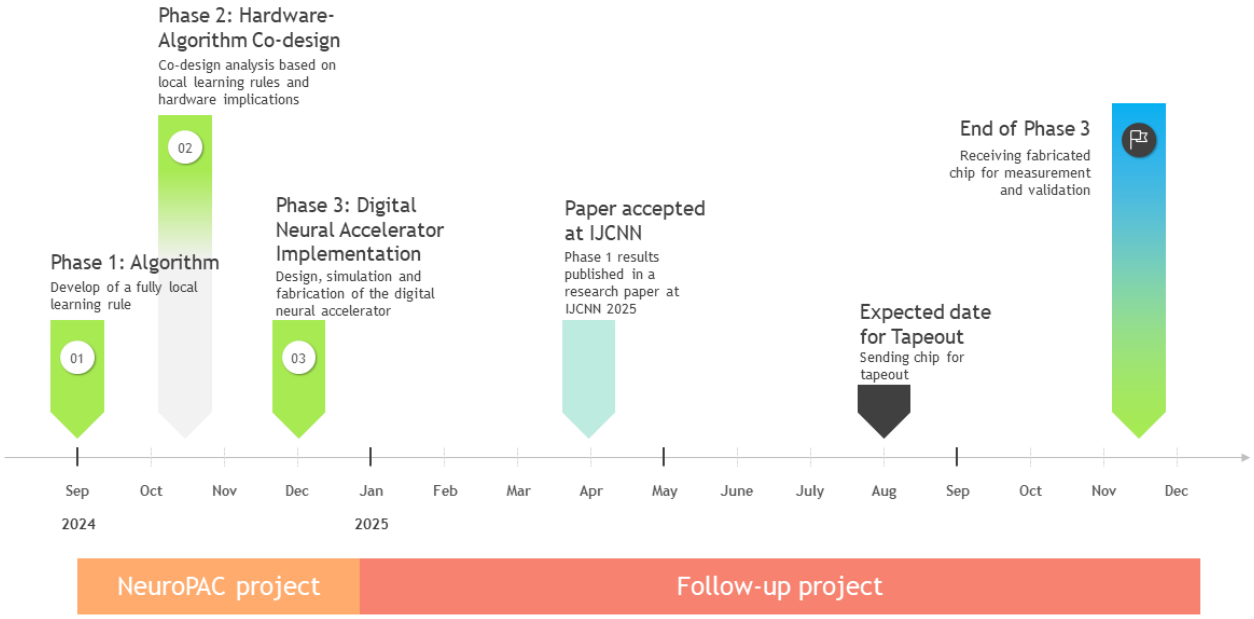


Figure 1: Project timeline. The project is structured into three clearly defined phases. Phases 1 (Algorithm Development) and 2 (Hardware-Algorithm Co-design) were both initiated and completed during the period supported by the NeuroPAC project. Phase 3 (Digital Neural Accelerator Implementation) began within the NeuroPAC timeframe and will continue as part of a longer-term follow-up project, which is expected to conclude by the end of 2025.

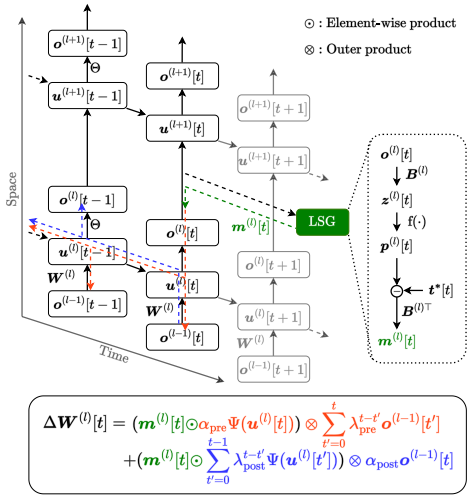


Figure 2: Overview of TESS. The diagram illustrates an SNN model unrolled in time, where $\boldsymbol{u}^{(l)}[t]$ denotes the membrane potential of neurons in the $l$-th layer at time step $t$, and $\boldsymbol{o}^{(l)}[t]$ represents the corresponding output spikes. Signals involved in weight update computation are highlighted: red represents the eligibility trace based on causal relationships between inputs and outputs, blue represents the eligibility trace for non-causal relationships, and green represents the local learning signal $\boldsymbol{m}^{(l)}[t]$ used to modulate the eligibility traces. The local learning signal is generated independently for each layer through a learning signal generation (LSG) process. The fixed binary matrix $\boldsymbol{B}^{(l)}$ used in the LSG process features columns corresponding to square wave functions. While, $f(\cdot)$ is a softmax function, and $\boldsymbol{t}^*[t]$ represent the labels.

on neuron-local information. As a result, its memory and computational overheads scale linearly with the number of neurons, independent of the sequence length.

TESS is particularly well-suited for real-time, on-device learning in low-power hardware due to its efficiency and scalability. It operates as a three-factor learning rule, minimizing computational and memory requirements. As illustrated in Fig. 2, TESS relies on two core components to achieve spatiotemporal locality: eligibility traces and locally generated learning signals. For full implementation details, experimental settings, and comparisons with prior work, please refer to the Appendix, which includes the manuscript accepted at the International Joint Conference on Neural Networks (IJCNN) 2025.

Despite its local nature, TESS achieves competitive performance with backpropagation through time (BPTT) on several benchmarks, including temporally encoded versions of CIFAR10 and CIFAR100, as well as the IBM DVS Gesture dataset. On CIFAR10-DVS, it incurs only a $\sim 1.4$ percentage point drop in accuracy. Importantly, TESS delivers substantial efficiency gains, reducing multiply-accumulate operations (MACs) by $205 - 661\times$ and memory usage by $3 - 10\times$, as summarized in Table 1.

Table 1: Comparison of performance and computational requirements of different local and non-local learning strategies on image classification tasks

| **Method** | **Model** | **Local Learning** | **Time-steps ($T$)** | **Batch Size** | **Accuracy[1] (mean±std)** | **# MAC[2] ($\times 10^6$)** | **Memory[2] (MB)** |
|---|---|---|---|---|---|---|---|
| | | | CIFAR10-DVS | | | | |
| **BPTT (baseline)** | VGG-9 | No | 10 | 64 | $76.40 \pm 0.66\%$ | 13589.59 | 25.50 |
| **S-TLLR (baseline)** | VGG-9 | Partial (in time) | 10 | 64 | $75.14 \pm 1.37\%$ | 13589.59 | 5.10 |
| **TESS (ours)** | VGG-9 | Yes | 10 | 64 | $75.00 \pm 0.65\%$ | 22.15 | 2.55 |
| | | | IBM DVS Gesture | | | | |
| **BPTT (baseline)** | VGG-9 | No | 20 | 16 | $97.95 \pm 0.68\%$ | 12079.69 | 22.69 |
| **S-TLLR (baseline)** | VGG-9 | Partial (in time) | 20 | 16 | $98.48 \pm 0.37\%$ | 12079.69 | 2.26 |
| **TESS (Ours)** | VGG-9 | Yes | 20 | 16 | $98.56 \pm 0.31\%$ | 22.65 | 2.26 |
| | | | CIFAR10 | | | | |
| **BPTT (baseline)** | VGG-9 | No | 6 | 128 | $92.55 \pm 0.06\%$ | 3623.90 | 6.83 |
| **S-TLLR (baseline)** | VGG-9 | Partial (in time) | 6 | 128 | $91.88 \pm 0.28\%$ | 3623.90 | 2.27 |
| **TESS (ours)** | VGG-9 | Yes | 6 | 128 | $92.55 \pm 0.16\%$ | 5.48 | 2.27 |
| | | | CIFAR100 | | | | |
| **BPTT (baseline)** | VGG-9 | No | 6 | 128 | $69.28 \pm 0.37\%$ | 3624.18 | 6.83 |
| **S-TLLR (baseline)** | VGG-9 | Partial (in time) | 6 | 128 | $68.00 \pm 0.71\%$ | 3624.18 | 2.27 |
| **TESS (ours)** | VGG-9 | Yes | 6 | 128 | $70.00 \pm 0.34\%$ | 17.64 | 2.27 |

[1]: Previous studies' accuracy values are provided as reported in their respective original papers.
[2]: # MAC and Memory are estimated for a batch size of 1.

### 2.2 Hadware-Algorithm Co-design Process with Local Learning Rules

As part of our co-design methodology, we reviewed prior art on on-device learning evaluated using the IBM DVS Gesture dataset. Table 2 summarizes the most relevant approaches. Among these, the highest reported accuracy (94.3%) was achieved by Chen et al. (2024) using a temporally local method (FPTT) combined with spatial credit assignment via backpropagation (BP). In contrast, their implementation using a fully local spatial method based on direct feedback alignment (DFA) attained a slightly lower accuracy (90.2%), demonstrating the trade-off between biological plausibility and task performance.

For a fair comparison and to isolate the contribution of our learning rule, we adopted the same model architecture used in Chen et al. (2024), which we denote as SmallConv. This spiking network comprises three convolutional layers (32C3-32C3-32C3), each with 32 output channels and a kernel size of $3 \times 3$, followed by a fully connected output layer with 11 neurons corresponding to the classification labels.

Table 2: Previous works focus on the on-device learning scenario evaluated on DVS Gesture dataset (Amir et al., 2017).

| Work | Reported Accuracy | Input Size | Learning Strategy | Model Architecture |
|---|---|---|---|---|
| Wong et al. (2021) | 83.0 % | – | Delta STDP (Transfer Learning) | – |
| Frenkel & Indiveri (2022) | 87.3 % | $16 \times 16$ | Modified Stochastic E-prop | 256FC-(256REC)-10FC |
| Zhang et al. (2023) | 92.0 % | $16 \times 16$ | Sparse Target Propagation | 1024FC-512FC-10FC |
| Chen et al. (2024) | 90.2 % | $32 \times 32$ | FPTT+DFA | 32C3-32C3-32C3-128FC10 |
| Chen et al. (2024) | 94.3 % | $32 \times 32$ | FPTT+BP | 32C3-32C3-32C3-128FC10 |

Next, we investigated the impact of quantization on the learning process, an essential step for hardware deployment. All computations in both the forward pass and weight update path were restricted to integer or fixed-point arithmetic to reduce the footprint of the downstream digital hardware architecture. In particular, all variables including weights, membrane potentials, and weight updates were quantized.

For weight quantization, we adopted a symmetric linear scheme, which has been successfully used for quantizing SNN models (Apolinario et al., 2023):

$$s_{\boldsymbol{W}} = \frac{\max(|\boldsymbol{W}_{\text{int}}|)}{2^{b-1}-1}, \tag{1}$$

$$\boldsymbol{W}_{\text{int}} = \text{clamp}\left(\text{round}\left(\frac{\boldsymbol{W}}{s_{\boldsymbol{W}}}\right), -2^{b-1}, 2^{b-1}-1\right), \tag{2}$$

where $s_{\boldsymbol{W}}$ is a scaling factor computed once at the beginning of training and kept constant, $\boldsymbol{W}_{\text{int}}$ denotes the quantized integer weights, and $b$ is the number of bits used for precision. The real-valued weights are approximated by

$$\boldsymbol{W} \approx s_{\boldsymbol{W}} \boldsymbol{W}_{\text{int}}. \tag{3}$$

To mitigate rounding artifacts, especially in low-precision regimes, we implemented stochastic rounding for the weight updates. The updates are quantized as

$$\Delta \boldsymbol{W}_{\text{int}} = \text{stochastic\_round}\left(\frac{\Delta \boldsymbol{W} \eta}{s_{\boldsymbol{W}}}\right), \tag{4}$$

where $\eta$ is the learning rate. To avoid the overhead of multiplication during hardware implementation, we selected $\eta$ as a power-of-two, allowing for efficient shift operations.

The weight update term $\Delta \boldsymbol{W}$ is computed based on the TESS learning rule shown in Fig. 2, omitting the non-causal terms highlighted in blue to reduce memory overhead. Additionally, we implemented hardware-friendly versions of the nonlinearities $\text{f}(\cdot)$ and $\Psi(\cdot)$ using piecewise linear approximations. Specifically,

$$\text{f}(x) = \begin{cases} 0 & \text{if } x \leq -2 \\ 0.25x + 0.5 & \text{if } -2 < x < 2 \\ 1 & \text{if } x \geq 2 \end{cases} \tag{5}$$

$$\Psi(x) = \max(1 - |x - c|, 0), \tag{6}$$

where $c$ is a tunable center parameter that defines the peak of the triangular function.

We fixed the membrane potential precision to 16 bits and explored different weight precisions (8, 10, and 12 bits). The performance under these configurations is reported in Table 3. Remarkably, we found that even at 8-bit precision, TESS achieved performance exceeding the baseline accuracy of 94.3% reported by Chen et al. (2024), despite using a fully local learning rule. This highlights the robustness and hardware efficiency of TESS, reinforcing its suitability for deployment in resource-constrained neuromorphic systems.

Table 3: Ablation studies on the impact of weight precision on the performance of a SmallConv model (≈20k parameters) on the DVS gesture dataset across different learning rates. Mean and standard deviation reported over 5 independent runs.

| $\eta$ | $b$=**8 bits** | $b$=**10 bits** | $b$=**12 bits** |
|---|---|---|---|
| 8 | $94.17 \pm 1.06$ | $94.55 \pm 0.79$ | $94.24 \pm 0.94$ |
| 4 | $94.77 \pm 0.82$ | $95.68 \pm 0.69$ | $95.00 \pm 0.49$ |
| 2 | $94.77 \pm 1.08$ | $94.70 \pm 0.97$ | $95.08 \pm 0.89$ |
| 1 | $93.41 \pm 0.79$ | $95.23 \pm 0.43$ | $95.23 \pm 0.69$ |

### 2.3 Hardware Architecture Design Overview

Based on our co-design analysis, we estimate the resource requirements for implementing the SmallConv model and the TESS learning rule. These are summarized in Table 4.

Table 4: Layer-wise resources table for a 3-layer fully connected network.

| **Layer** | **Input Size** | **# Params** | **Weights (Bytes)** | **Membrane (Bytes)** | **Input (Bits)** | **Output (Bits)** | **Trace (Bytes)** |
|---|---|---|---|---|---|---|---|
| Conv3x3 | 2x2x32 | 576 | 576 | 65536 | 8192 | 32768 | 4096 |
| MaxPool | 32x32x32 | - | - | - | 32768 | 8192 | - |
| Conv3x3 | 32x16x16 | 9216 | 9216 | 16384 | 8192 | 8192 | 16384 |
| MaxPool | 32x16x16 | - | - | - | 8192 | 2048 | - |
| Conv3x3 | 32x8x8 | 9216 | 9216 | 4096 | 2048 | 2048 | 4096 |
| AvgPool | 32x8x8 | - | - | - | 2048 | 640 | - |
| Linear | 128 | 1408 | 1408 | - | 640 | 17 | - |

These requirements inform the design of the full hardware architecture, including memory address allocation and control logic. Figure 3 illustrates the architecture for a single network layer, which we describe below, as all layers follow a similar structure. The architecture consists of three main components: the Forward Propagation Block, the Learning Signal Generation Block, and the Eligibility Trace Block.

Importantly, this architecture benefits directly from the spatiotemporal locality of TESS. Since all computations rely solely on neuron-local information and avoid global gradient propagation, each layer operates independently and requires only local memory and interconnects. This drastically simplifies control logic and data movement, resulting in a highly modular and potentially energy-efficient design. By avoiding the need for global error signals or long-range memory fetches, the TESS-based hardware is expected to achieve significant reductions in area, latency, and power—making it well-suited for on-device learning in edge systems.

These requirements inform the design of the full hardware architecture, including memory address allocation and control logic. Figure 3 illustrates the architecture for a single network layer. Since all layers share a similar structure, we describe a single-layer instantiation below. The architecture consists of three main components:

**Forward Propagation Block:** This block implements the leaky integrate-and-fire (LIF) neuron dynamics. It receives and decodes input spikes from the event register, updates the membrane potentials of neurons, and emits output spikes if thresholds are exceeded. It is active during both inference and learning phases.

**Learning Signal Generation (LSG) Block:** This module computes the modulatory signal $\boldsymbol{m}[t]$ as shown in Fig. 2. It is only active during the learning phase and corresponds to the top-down error-modulated signal in TESS.

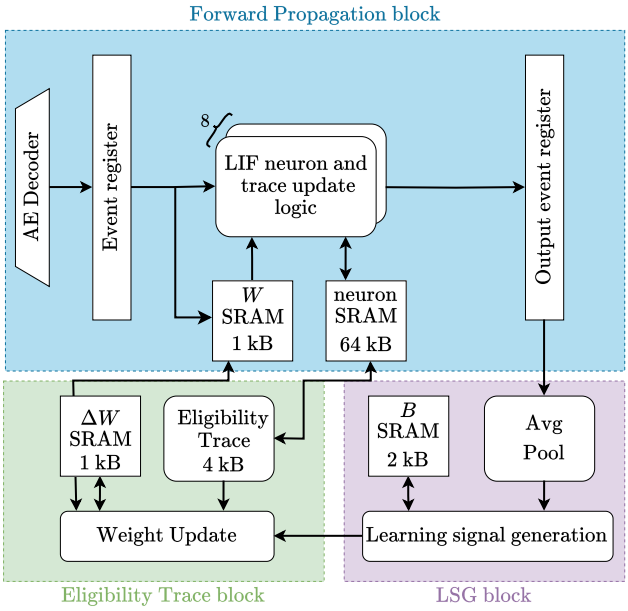


Figure 3: Overview of a hardware architecture supporting the TESS learning rule.

**Eligibility Trace Block:** This block computes the eligibility trace based on causal spike interactions (red path in Fig. 2) and combines it with the learning signal to compute the final weight updates $\Delta \boldsymbol{W}$.

## 3 Outcomes and Next Steps

This work aimed to co-design a biologically inspired and hardware-friendly local learning rule for SNNs. Toward this goal, we developed TESS, a novel Temporally and Spatially Local Learning Rule for SNNs, and explored its algorithmic and hardware characteristics.

On the algorithmic side, we evaluated TESS under varying weight precisions to assess its robustness to quantization, an essential consideration for hardware deployment. These experiments confirmed that TESS maintains strong performance even at reduced precision, reinforcing its suitability for memory- and energy-constrained systems. The core learning rule and its algorithmic performance are presented in a paper accepted at the International Joint Conference on Neural Networks (IJCNN) 2025, which is attached in the appendix and publicly available as a preprint at arXiv:2502.01837. To support open science and reproducibility, we have also released the source code at github.com/mapolinario94/TESS.

On the hardware side, we designed a digital architecture supporting on-device learning with TESS. The architecture uses only local information at both spatial and temporal scales, reducing the communication and memory bottlenecks typical of global learning methods like BPTT. We completed the Verilog implementation and simulation of the *forward propagation* and *learning signal generation* blocks. Implementation of the *eligibility trace* block is ongoing.

Next steps include (covered under the ongoing Phase 3 in the project timeline, see Fig. 1):

- Finalizing the Verilog implementation of the *eligibility trace* block,
- Verifying system-wide functionality through testbench simulations,
- Preparing for tapeout and hardware testing in a real-world setting.

These efforts contribute toward the long-term goal of enabling energy-efficient, on-device learning in neuromorphic systems using principled, scalable, and low-footprint local learning rules.

## References


Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-January, pp. 7388–7397. IEEE, 7 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.781.

Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Sapan Agarwal, Matthew Marinella, Martin Foltin, John Paul Strachan, Dejan Milojicic, Wen Mei Hwu, and Kaushik Roy. PANTHER: A Programmable Architecture for Neural Network Training Harnessing Energy-Efficient ReRAM. *IEEE Transactions on Computers*, 69(8):1128–1142, 8 2020. ISSN 15579956. doi: 10.1109/TC.2020.2998456.

Marco P. E. Apolinario, Adarsh Kumar Kosta, Utkarsh Saxena, and Kaushik Roy. Hardware/Software co-design with ADC-Less In-memory Computing Hardware for Spiking Neural Networks. *IEEE Transactions on Emerging Topics in Computing*, pp. 1–13, 2023. ISSN 2168-6750. doi: 10.1109/TETC.2023.3316121. URL `https://ieeexplore.ieee.org/document/10260275/`.

Faquan Chen, Qingyang Tian, Lisheng Xie, Yifan Zhou, Ziren Wu, Liangshun Wu, Rendong Ying, Fei Wen, and Peilin Liu. EPOC: A 28-nm 5.3 pJ/SOP Event-driven Parallel Neuromorphic Hardware with Neuromodulation-based Online Learning. *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–16, 2024. ISSN 1932-4545. doi: 10.1109/TBCAS.2024.3470520.

Charlotte Frenkel and Giacomo Indiveri. ReckOn: A 28nm Sub-mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales. In *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, volume 2022-February, pp. 468–470. Institute of Electrical and Electronics Engineers Inc., 2022. ISBN 9781665428002. doi: 10.1109/ISSCC42614.2022.9731734.

Xiaochen Peng, Shanshi Huang, Hongwu Jiang, Anni Lu, and Shimeng Yu. DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(11):2306–2319, 11 2021. ISSN 19374151. doi: 10.1109/TCAD.2020.3043731.

M. M. Wong, S. B. Shrestha, V. P. Nambiar, A. Mani, Y. K. Lee, E. K. Koh, W. Jiang, K. T. C. Chai, and A. T. Do. A 2.1 pJ/SOP 40nm SNN Accelerator Featuring On-chip Transfer Learning using Delta STDP. In *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, pp. 95–98. IEEE, 9 2021. ISBN 978-1-6654-3751-6. doi: 10.1109/ESSCIRC53450.2021.9567782.

Jilin Zhang, Dexuan Huo, Jian Zhang, Chunqi Qian, Qi Liu, Liyang Pan, Zhihua Wang, Ning Qiao, Kea-Tiong Tang, and Hong Chen. 22.6 ANP-I: A 28nm 1.5pJ/SOP Asynchronous Spiking Neural Network Processor Enabling Sub-O.1 $\mu$J/Sample On-Chip Learning for Edge-AI Applications. In *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 21–23. IEEE, 2 2023. ISBN 978-1-6654-9016-0. doi: 10.1109/ISSCC42615.2023.10067650.

Qingtian Zhang, Huaqiang Wu, Peng Yao, Wenqiang Zhang, Bin Gao, Ning Deng, and He Qian. Sign backpropagation: An on-chip learning algorithm for analog RRAM neuromorphic computing systems. *Neural Networks*, 108:217–223, 12 2018. ISSN 0893-6080. doi: 10.1016/J.NEUNET.2018.08.012.