PINSim: A Processing In- and Near-Sensor Simulator to Model Intelligent Vision Sensors

Sepehr Tabrizchi, *Student Member, IEEE*, Mehrdad Morsali, *Student Member, IEEE*, David Pan, *Fellow, IEEE*, Shaahin Angizi, *Senior Member, IEEE*, and Arman Roohi, *Senior Member, IEEE*

***Abstract*—This letter introduces PINSim, a user-friendly and flexible framework for simulating emerging smart vision sensors in the early design stages. PINSim enables the realization of integrated sensing and processing near and in the sensor, effectively addressing challenges such as data movement and power-hungry analog-to-digital converters. The framework offers a flexible interface and a wide range of design options for customizing the efficiency and accuracy of processing-near/in-sensor-based accelerators using a hierarchical structure. Its organization spans from the device level upward to the algorithm level. PINSim realizes instruction-accurate evaluation of circuit-level performance metrics. PINSim achieves over $25,000\times$ speed-up compared to SPICE simulation with less than a 4.1% error rate on average. Furthermore, it supports both multilayer perceptron (MLP) and convolutional neural network (CNN) models, with limitations determined by IoT budget constraints. By facilitating the exploration and optimization of various design parameters, PiNSim empowers researchers and engineers to develop energy-efficient and high-performance smart vision sensors for a wide range of applications.**

***Index Terms*—Processing-in-sensor, processing-near-sensor, numerical simulation, neural network.**

I. Introduction

EDGE Artificial Intelligence (AI) has rapidly evolved into an integral part of modern technological ecosystems, fundamentally reshaping how we process data and interact with our digital environment. The significance of this technology cannot be overstated, as it pushes the boundaries of traditional cloud-based AI, moving towards more localized, low-latency, and power-efficient solutions. This transformation is primarily driven by the surge in Internet of Things (IoT) devices, which generate vast amounts of data. However, nearly 90% of this data remains unanalyzed due to the limited computing capabilities of current IoT devices and the compute-intensive nature of the convolutional neural network (CNN) algorithms [1].

To address these challenges, computing architectures must transition from a cloud-centric to a thing-centric (data-centric) approach, where IoT nodes process the sensed data locally. Processing-Near-Sensor (PNS) architectures have been extensively studied as a potential solution, where an on-chip processor accelerates digital pixel outputs near the sensor. Inspired by Processing-in-Memory (PIM) architectures, these paradigms aim to overcome the bottlenecks of von Neumann computing models, such as lengthy memory access latency and energy-intensive data transfers. Smart image sensors with preprocessing capabilities [2], [3] represent a significant step forward in this direction. Building on PNS and PIM techniques, Processing-in-Sensor (PIS) units operate on pre-analog-to-digital converter (pre-ADC) data, enhancing vision sensor functionality and eliminating redundant data output [4], [5]. Hybrid PIS-PNS platforms [6] combine the strengths of both approaches, further improving data processing speed and reducing power consumption. In [7], all computations are performed in the analog domain. The authors in [8] replaced ADC with a novel component to implement the spike neural network (SNN) within the sensor. The Processing-in-Pixel (PIP) paradigm has recently emerged as another promising approach within the rapidly developing fields of near-/in-sensor and in-memory computing platforms. This paradigm processes data pre-ADC before converting and transmitting it to the on/off-chip processor [9], [10]. By processing data locally, PIP significantly reduces power consumption, accelerates data processing, and alleviates the memory bottleneck problem [6]. Due to PIP's limited resources, deploying all neural network layers within the pixel array is inefficient. Consequently, most studies have accelerated the first layers in an analog domain, submitting the remaining layers to digital neural network accelerators. For example, RedEye [11] implements convolution operations using charge-sharing tunable capacitors, sacrificing accuracy for energy savings. Similarly, MACSen [2] processes the first convolutional layer of binary-weight neural networks using the correlated double sampling method at 1000fps. However, it suffers from significant area overhead and high power consumption due to its SRAM-based design. Another PIP-based architecture [9] operates convolution in an analog domain to reduce power consumption but requires four photodiodes and four capacitors per pixel, resulting in area and power overheads. Given the distinct workload sizes and memory access patterns, the PIP paradigm is poised to benefit various image processing applications at the edge. However, selecting the appropriate design for a specific application is vital and challenging. Furthermore, establishing uniform evaluation conditions is imperative for making impartial choices between available design options.

Despite significant progress, there remains a lack of behavior-/system-level performance and energy evaluation frameworks for PNS/PIS architectures. Researchers have developed various in-house memory evaluation tools, such as Cacti [12] and NVSIM [13] to facilitate design space exploration and optimization of memory systems. Analog and digital PIM accelerator simulators like NeuroSim [14] and PIMSim [15], respectively,

Received 24 October 2024; accepted 23 December 2024. Date of publication 25 December 2024; date of current version 17 January 2025. This work was supported in part by the National Science Foundation under Grant 2448133, Grant 2216772, Grant 2504839, and Grant 2216773, and in part by the Semiconductor Research Corporation (SRC). *(Corresponding author: Arman Roohi.)*

Sepehr Tabrizchi and Arman Roohi are with the Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago, IL 60607 USA (e-mail: aroohi@uic.edu).

Mehrdad Morsali and Shaahin Angizi are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: shaahin.angizi@njit.edu).

David Pan is with the Department of Electrical and Computer Engineering, University of Texas Austin, Austin, TX 78712 USA (e-mail: dpan@ece.utexas.edu).

Digital Object Identifier 10.1109/LCA.2024.3522777

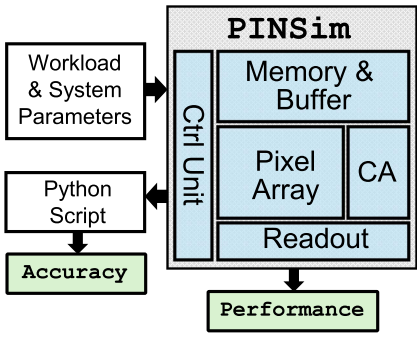

Fig. 1. The proposed PINSim Framework.

have also been developed to predict performance and energy consumption reliably. This paper introduces a Processing In-and Near-Sensor simulator (PINSim) to model intelligent vision sensors, providing a flexible framework for designing emerging PNS, PIS, and PIP architectures and evaluating their performance. The PINSim source codes are publicly available on the Github.[1]

II. PINSim Framework

The PINSim system takes user requirements and inputs, such as target energy and hardware limits. It produces a close estimate of performance metrics and accuracy, as shown in Fig. 1. PINSim utilizes device/circuit features extracted offline using Cacti, NVSIM, and NVMain. It first uses the workloads and system settings to create a PNS/PIS/PIP structure. A separate Python script external to PINSim simultaneously quantizes the first layer's neural network weights. The neural network is modified based on the quantized weights, and its accuracy is examined. Since the visionary IoT has limited resources and full implementation of neural networks is computationally intensive, we only focus on the first layer, and analog/digital deep-learning accelerators will calculate other layers. The given settings determine the buffer sizes and the number of compute add-ons (CAs). If any input is missing, the default value is used. After getting all the needed parts, e.g., pixel array size etc., the proper control unit, including row and column decoders, is considered, and all the parts are assembled. To lower the chance of errors or/and design problems, the functionality of the designed architecture is checked through several if/else conditions. Since this is a recurring process, it continues until the desired error-free design is achieved. If all the design criteria are met, high-level settings are assessed to provide the system's performance, including delay, area, power, and frame rate.

A. Modular Approach

PINSim encapsulates device-upwards system modeling by integrating different memory technologies, pixel designs, and various in-sensor and near-sensor processing paradigms. Its modular architecture allows users to easily add/remove and/or enable/disable components and various event detection or object classification algorithms. The flexibility of PINSim facilitates the exploration and evaluation of intelligent vision sensor systems across a wide range of configurations and use cases. It is important to note that the resulting design is not the best-optimized design, and users could consider more optimization steps, including: a) Considering the use of more advanced interconnect techniques or 3D stacking; b) Exploring the addition of extra components, such as a microlens array on top of the pixel array; c) Exploring the use of more efficient pixel design; d) Considering the adoption of a pipelined ADC architecture; e) Investigating the potential of using a time-interleaved ADC architecture or replacing the current ADC design with a more efficient one like Successive Approximation Register (SAR) ADC; f) Fine-tuning the RC network; g) Implementing an adaptive clock rate. One of PINSim's key advantages is its user-friendly plug-and-play capability, which allows users to add or modify the specified building blocks easily. Please note that PINSim is a system-level simulation engine and does not offer built-in, fine-grained optimization techniques for every individual component.

[1][Online]. Available: https://github.com/iDEALabAcademy/PINSim

TABLE I
Validation Results for PINSim Operations At 45nm

Design	Tool	Sense Power (W)	Sense Latency (s)	Compute Power (W)	Compute Latency (s)
MLP	SPICE	1.833e-03	2.850e-05	1.328e+01	1.563e-03
	PINSim	1.902e-03	2.800e-05	1.375e+01	1.467e-03
	Error Rate	-3.6%	+1.8%	-3.4%	+6.6%
CNN	SPICE	4.614e-06	2.847e-05	9.655e-02	1.804e-02
	PINSim	4.752e-06	2.800e-05	1.010e-01	1.681e-02
	Error Rate	-2.9%	+1.7%	-4.4%	+7.4%

B. Supported Operation Modes

1) Sensing: In the sensing mode, PINSim functions akin to a conventional image sensor. Initially, pixels are turned into inverse polarization in sensing mode, allowing capacitors to charge fully. Then, concerning the external light intensity, a photo-current is produced, and the voltage values before and after the image light exposure are sampled. The difference between the two voltages is sensed with an amplifier. It is worth pointing out that each ADC sample subtracts the pixel reset voltage when the voltage drops and converts the output signal. Accordingly, the ADC can skip to the next row of the array.

2) Processing. Event Detection: Primarily operating in object-detection mode, the system employs a novel approach by grouping pixels into $x \times x$ boxes, wherein only the central pixel remains active. The accuracy of ADCs is deliberately reduced by deactivating the fine circuit, a strategy aimed at optimizing processing efficiency. A significant feature of this mode is comparing central pixels against their preceding states to detect variations; upon detecting a change, the entire box is activated. PINSim supports three background-subtraction approaches to detect an event, including static frame difference, consecutive frame difference, and the hybrid frame difference scheme positioned in between the first conventional two methods. In object-detection mode, PINSim is adept at performing the initial layer of the network internally within the sensor. The image sensor incorporates one of the modified pixel architectures with CAs repositioned closer to the pixel array. The count of these add-ons is optimized to $m \times n$, correlating with the dimensions of the pixel array.

Object Classification: The final mode is object classification, where a reconfigurable processing framework within the sensor is utilized. This design signifies PINSim's capability for all three processing near/in-sensor and in-pixel, marking a significant advancement in sensor technology. The architecture is capable of

processing the initial layer of either multilayer perceptron (MLP) or convolutional neural network (CNN). A notable feature is the reconfig urable ADC, which facilitates the near or in-sensor implementation of activation functions such as ReLU and sign functions.

C. Performance Modeling

Established simulators often rely on published current values measured on physical devices, posing a challenge when such data is unavailable for emerging architectures and prototype designs. To overcome this limitation, PINSim incorporates a device-level energy model that provides independent power calculations for various modules within the system. This allows for the utilization of values readily obtained from circuit-level simulators like NVSIM or CACTI, with each operation incrementally updating the system's energy usage. Standby and power-down energies are calculated using simulated leakage results, enabling a comprehensive energy consumption analysis even without published device data.

D. Neural Network Flexibility

To the best of our knowledge, no comprehensive simulator currently effectively models PNS, PIS, and PIP architectures. Our previous simulator [16] was limited to modeling PIP architectures and only supported CNNs. In contrast, PINSim is designed to model all three types of discussed architectures and supports both CNNs and MLP networks. It is important to note that the attributes of the first layer, including the spatial dimensions of the input feature map (ifmap) and filter, as well as the number of channels, are constrained due to IoT limitations. However, users have the flexibility to adjust these input matrices beyond the default limits by modifying the code.

E. Behavior-Level Accuracy Model

To validate the correctness of modified neural networks and ensure acceptable accuracy, we developed a Python script that runs above the PyTorch library and allows users to apply their modifications. This script can be run parallel with the performance models. The script takes several inputs, such as a pre-trained model, a target dataset, and user-specified parameters. The key parameters affecting accuracy are the quantization level, filter size, channel pruning, pixel array size, and stride. Since the PINSim framework focuses on the first layer, this script can operate in parallel. The first layer should only be adjusted using the defined inputs, while the rest remain unchanged. Regarding filters of the first layer, the quantization level variable forces the script to quantize the weights properly.

III. Case Studies

A. Verification Results

To ensure the accuracy of PINSim in reporting power and latency parameters, we implement the complete architecture, including the peripherals, using the Synopsys HSPICE simulator. The simulations were executed on a system equipped with 128 GB of RAM and an AMD EPYC 7302P 16-core processor running SUSE Linux. State-of-the-art designs for PNS, PIS, and PIP architectures were not suitable for validation due to the lack of necessary detailed simulation parameters and configurations.

TABLE II
Simulation Time and Memory Utilization of PINSim Vs. SPICE

Size	Networks	PINSim (second)	SPICE (second)	Speed Up	PINSim (MB)	SPICE (MB)	Memory Saving
32x32	CNN	0.025	53.71	2148	15.026	578.88	38
	MLP	0.026	124.21	4777	15.025	1201.68	79
64x64	CNN	0.031	171.86	5543	15.035	851.17	56
	MLP	0.03	3191.74	106391	15.035	2015.69	134
128x128	CNN	0.031	667.31	21526	15.038	1868.12	124
	MLP	0.031	3761.58	121341	15.034	6684.45	444
256x256	CNN	0.033	2621.69	79445	15.041	6201.32	412
	MLP	0.034	8452.23	248595	15.038	16245.52	1080

Therefore, we implemented the MLP and CNN designs with a 3×3 box size in both PINSim and SPICE at the 45nm PTM technology node, with a nominal V_{DD} of 1V. The results, classified into sensing and computation components, are presented in Table I. We found that the error rates for latency and power consumption were within 7.4% and 4.4%, respectively. Additionally, the table highlights a difference in power consumption during sensing between MLP and CNN. This disparity stems from the larger memory size in MLP, which results in higher static memory power consumption and significantly influences the system's total power consumption.

While SPICE circuit simulators provide exceptional accuracy and flexibility, the main limitation of PINSim is specifically designed for PNS, PIS, and PIP-based architectures. Moreover, PINSim offers a user-friendly interface with significantly higher simulation speeds. To evaluate this, we implemented the complete architectures for MLP and CNN in SPICE and compared the simulation time and memory utilization of PINSim against SPICE, as presented in Table II. To further assess the effect of the pixel array size in PINSim (i.e., 32×32, 64×64, 128×128, and 256×256) on the overall simulation time, we conducted several simulations using a box size of 3×3. The results, including peak memory usage and total CPU time, are reported. In contrast, PINSim quickly estimates performance metrics by leveraging analytical models and/or predefined values provided by the user, while HSPICE solves multiple equations. Instead of solving these equations, PINSim utilizes pre-calculated component characteristics to compute performance metrics such as power consumption. We observed that in CNN architecture, PINSim achieves, on average, a $25,000\times$ speed-up compared to SPICE. Additionally, larger pixel array sizes result in even more significant speed-ups with PINSim. In terms of memory usage, PINSim requires, on average, $158\times$ less memory to store and generate the results compared to SPICE.

B. Validation Results

Herein, we utilize the PINSim simulator to validate its capability to accurately represent emerging smart vision sensor architectures across five different designs, which are labeled as Design1 (D1) through Design4 (D4). Another design, D5, closely resembles D3, with the key difference that the CAs are positioned closer to the pixels. Table III provides a detailed comparison of all these designs. To ensure a consistent comparison, most parameters are kept constant across all examples, highlighting the impact of varying parameters on the results. Each design is simulated under 45 nm technology with an array size of 32×32 and a box size of 3×3. The weight precision is set to 8-bit, and the models for pixels, CAs, ADCs, memories, and controllers are identical across all scenarios. D1 and D2 explore architectures for MLP, while Examples D3, D4, and D5 focus on CNN architectures. In D2 and D4, the level of

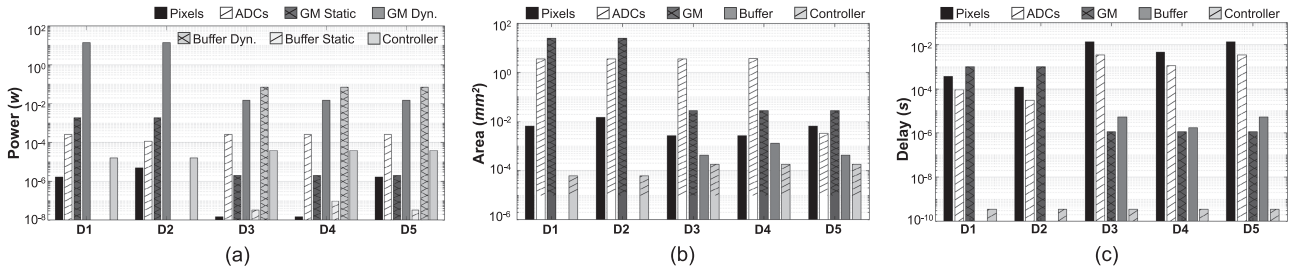

Fig. 2. Calculated (a) power consumption, (b) area, and (c) delay of the examined designs using PINSim.

TABLE III
DIFFERENT DESIGNS' PARAMETERS

Configuration	D1	D2	D3	D4
Total_power	4.13E+00	4.13E+00	3.06E-02	3.06E-02
total_delay	1.47E-03	1.17E-03	1.68E-02	5.61E-03
total_area	2.86E+01	2.86E+01	3.60E+00	3.71E+00
total_normal_delay	2.80E-05	2.80E-05	2.80E-05	2.80E-05
total_sensing_delay	9.63E-06	9.63E-06	9.63E-06	9.63E-06
computing_delay	1.47E-03	1.17E-03	1.68E-02	5.61E-03
FPS_normal	3.57E+04	3.57E+04	3.57E+04	3.57E+04
FPS_sensing	1.04E+05	1.04E+05	1.04E+05	1.04E+05
FPS_computing	6.82E+02	8.56E+02	5.95E+01	1.78E+02
total_adcs_in_compute	1	3	11	33
total_adcs_in_sensing	11	11	11	11
total_adcs_in_normal	32	32	32	32
Global memory_size	512.00 KB	512.00 KB	576.00 Bytes	576.00 Bytes
Buffer memory_size	0	0	9.00 Bytes	27.00 Bytes

parallelism is increased to three, indicating the presence of three separate computational paths. As shown in Table III, the power consumption in D2 and D4 exceeds that of D1 and D3 due to the increased number of CAs and ADCs, which also results in significantly higher speed. The memory configuration remains constant, leading to equal values for global memory across D1 with D2 and D3 with D4, as depicted in the table. Notably, in MLP architectures, no buffer memory is required.

Fig. 2 presents the comparison results for D1 through D5 in terms of power, area, and delay with respect to individual components. Increasing parallelism in CNN architectures necessitates additional buffer memories to accommodate the kernel loading process. While in MLP architectures, it is optimal to position the CAs close to the pixels, in CNN architectures, it is more efficient to place them before the ADC. Comparing D3 and D5 illustrated this. Comparing D3 and D5, as shown in the figures, reveals no improvement in delay for D5 (Fig. 2(c)), while both the area and power consumption of the pixels have increased, shown in Fig. 2(a) and (b). Based on Fig. 2(a), reading data from the global memory (GM) consumes the most energy in the system. Additionally, fewer CAs make this architecture more compact and efficient. These factors highlight two key advantages of CNNs over MLPs: while MLP systems demonstrate faster processing speeds, CNNs are more power-efficient due to their lower resource requirements.

IV. CONCLUSION

PINSim offers a comprehensive framework for simulating intelligent vision sensor architectures, supporting PNS, PIS, and PIP paradigms. It provides accurate performance estimates with significant speed-ups over SPICE simulations while supporting both MLP and CNN architectures. The simulator's modular approach enables easy customization and exploration of design parameters. Case studies demonstrate that PINSim achieves over 25,000× speed-up compared to SPICE simulations with error rates below 7.4% for latency and 4.4% for power consumption.

REFERENCES

[1] K.-T. Tang et al., "Considerations of integrating computing-in-memory and processing-in-sensor into convolutional neural network accelerators for low-power edge devices," in *Proc. 2019 Symp. VLSI Technol.*, 2019, pp. T166–T167.
[2] H. Xu et al., "MACSen: A processing-in-sensor architecture integrating mac operations into image sensor for ultra-low-power BNN-based intelligent visual perception," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 68, no. 2, pp. 627–631, Feb. 2021.
[3] M. Abedin, A. Roohi, M. Liehr, N. Cady, and S. Angizi, "MR-PIPA: An integrated multi-level rram (HfOx) based processing-in-pixel accelerator," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 8, no. 2, pp. 59–67, Dec. 2022.
[4] S. Tabrizchi, S. Angizi, and A. Roohi, "TizBin: A low-power image sensor with event and object detection using efficient processing-in-pixel schemes," in *Proc. IEEE 40th Int. Conf. Comput. Des.*, 2022, pp. 770–777.
[5] T. Ma et al., "LeCA: In-sensor learned compressive acquisition for efficient machine vision on the edge," in *Proc. Annu. Int. Symp. Compu. Architecture*, 2023, pp. 1–14.
[6] T.-H. Hsu et al., "AI edge devices using computing-in-memory and processing-in-sensor: From system to device," in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 22–5.
[7] B. Zambrano, S. Strangio, T. Rizzo, E. Garzón, M. Lanuzza, and G. Iannaccone, "All-analog silicon integration of image sensor and neural computing engine for image classification," *IEEE Access*, vol. 10, pp. 94417–94430, 2022.
[8] Z. Li, Q. Zheng, Y. Chen, and H. Li, "SpikeSen: Low-latency in-sensor-intelligence design with neuromorphic spiking neurons," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 70, no. 6, pp. 1876–1880, Jun. 2023.
[9] R. Song, K. Huang, Z. Wang, and H. Shen, "A reconfigurable convolution-in-pixel CMOS image sensor architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 7212–7225, Oct. 2022.
[10] G. Datta et al., "A processing-in-pixel-in-memory paradigm for resource-constrained tinyml applications," *Sci. Rep.*, vol. 12, no. 1, 2022, Art. no. 14396.
[11] R. LiKamWa et al., "Redeye: Analog convnet image sensor architecture for continuous mobile vision," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 255–266, 2016.
[12] N. Muralimanohar et al., "Cacti 6.0: A tool to model large caches," HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2009-85, 2009.
[13] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.
[14] P.-Y. Chen et al., "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.
[15] S. Xu, X. Chen, Y. Wang, Y. Han, X. Qian, and X. Li, "PIMSim: A flexible and detailed processing-in-memory simulator," *IEEE Comput. Architecture Lett.*, vol. 18, no. 1, pp. 6–9, Jan.-Jun. 2019.
[16] A. Roohi, S. Tabrizchi, M. Morsali, D. Z. Pan, and S. Angizi, "PiPSim: A behavior-level modeling tool for CNN processing-in-pixel accelerators," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 43, no. 1, pp. 141–150, Jan. 2024.