Abstract—This work proposes a Processing-In-Sensor Accelerator, namely PISA, as a flexible, energy-efficient, and high-performance solution for real-time and smart image processing in AI devices. PISA intrinsically implements a coarse-grained convolution operation in Binarized-Weight Neural Networks (BWNMs) leveraging a novel compute-pixel with non-volatile weight storage at the sensor side. This remarkably reduces the power consumption of data conversion and transmission to an off-chip processor. The design is completed with a bit-wise near-sensor processing-in-DRAM computing unit to process the remaining network layers. Once the object is detected, PISA switches to typical sensing mode to capture the image for a fine-grained convolution using only the near-sensor processing unit. Our circuit-to-application co-simulation results on a BWNM acceleration demonstrate acceptable accuracy on various image datasets in coarse-grained evaluation compared to baseline BWNM models, while PISA achieves a frame rate of 1000 and efficiency of $\sim$1.74 TOps/W. Lastly, PISA substantially reduces data conversion and transmission energy by $\sim$84% compared to a baseline CPU-sensor design.

Index Terms—Processing-in-sensor, accelerator, magnetic memories.

I. INTRODUCTION

INTERNET of Thing (IoT) devices are projected to attain an $1100B market by 2025, with a web of interconnection projected to comprise approximately 75+ billion IoT devices, including wearable devices, smart cities, and smart industry [1], [2]. Intelligent IoT (IIoT) nodes consist of sensory systems, which enable massive data collection from the environment and people to process with on-off-chip processors (10$^{19}$ bytes/s or ops). In most cases, large portions of the captured sensory data are redundant and unstructured. Data conversion and transmission of large raw data to a back-end processor impose high energy consumption, high latency, a memory bottleneck, and low-speed feature extraction on the edge [1], [3], [4] as shown with the pixel-only architecture in Fig. 1. To overcome these issues, computing architectures will need to shift from a cloud-centric approach to a thing-centric (data-centric) approach, where the IoT node processes the sensed data. Nonetheless, the processing demands of artificial intelligence tasks such as Convolutional Neural Networks (CNNs) spanning hundreds of layers face serious challenges for their tractability in computational and storage resources. Effective techniques in both software and hardware domains have been developed to improve CNN efficiency by alleviating the “power and memory wall” bottleneck.

In algorithm-based approaches, the use of shallower but wider CNN models, quantizing parameters, and network binarization has been explored thoroughly [5]–[7]. Recently, low bit-width weights and activations reduces computing complexity and model size. For instance, in [5], authors performed bit-wise convolution between the inputs and low bit-width weights by converting the conventional Multiplication-And-Accumulate (MAC) into the corresponding AND-bitcount operations. In an extreme quantization method, Binary Convolutional Neural Network (BCNN) has achieved acceptable accuracy on both small [8] and large datasets [7] by relaxing the demands for some high precision calculations. Instead, it binarizes weight and/or input feature map while processing the forward path, providing a promising solution to mitigate aforementioned bottlenecks in storage and computational components [2].

From the hardware point of view, the underlying operations should be realized using efficient mechanisms. However, the conventional processing elements are developed based on the von-Neumann computing model with separate memory and processing blocks connecting via buses, which imposes serious challenges, such as long memory access latency, limited memory bandwidth, energy-hungry data transfer, and high leakage power consumption restricting the edge device’s efficiency and working hours [2], [10]. Besides, in the upper level, this causes several significant issues such as communication bandwidth and security. Therefore, as a potential remedy, smart image sensors with instant image preprocessing have been extensively explored for object recognition applications [2], [11]–[13]. This paves the way for new sensor paradigms such as a Processing-Near-Sensor (PNS), in which digital outputs of a pixel are accelerated near the sensor leveraging an on-chip processor. Another solution to alleviate the above-mentioned challenges is a Processing-in-Memory (PIM) architecture, which is extensively studied in [10], [14]–[16]. By inspiring the PNS and PIM techniques, two promising alternatives are the Processing-in-Sensor (PIS) that works on pre- Analog-to-Digital Converters (ADC) data [12], [17] and a hybrid PIS-PNS platform [1] to improve vision sensor functionality and eliminate redundant data output, as shown in Fig. 1. However, the computational capabilities of these sensors have been limited to specific applications since enhancing throughput is followed by a growth in sensor temperature; and higher temperatures lead to noise that degrades sensing accuracy [18]. This includes specific feature extraction applications less supporting MAC-based image classification [1], [11] to meet both resiliency and efficiency such as Haar-like image filtering [19], sharpening, blurring [13], and local binary pattern [20].

In this paper, we propose a new Processing-In-Sensor Accelerator (PISA) as an energy-efficient PIS paradigm co-
integrating always-on sensing and processing capabilities working with a near-sensor PIM unit that is categorized as a new hybrid design. The proposed design features a real-time programmable coarse-grained convolution to reduce the power consumption of data conversion from photo-currents to pixel values in the image processing task. Once the object is detected, PISA switches to a typical sensing mode to capture the image for fine-grained convolution using a PNS unit. The contributions of this paper are as follows:

1) We develop a PIS architecture based on a set of innovative microarchitectural and circuit-level schemes optimized to process the 1st-layer of BWNNs with weights stored in non-volatile memory components that offers energy-efficiency and speed-up.
2) We complete the design with a bit-wise near-sensor PIM-enabled unit based on DRAM to process the remaining network layers. It leverages the charge-sharing feature of the DRAM cell and elevates it to implement the operation based on a dual-row activation mechanism.
3) We present a solid bottom-up evaluation framework and a PIM assessment simulator to analyze the performance of the whole system.
4) We extensively assess PISA’s performance and energy-efficiency co-integrated with the near-sensor PIM unit compared with recent sensory platforms.

The remainder of the paper is designed as follows. Section II discusses the state-of-the-art near-sensor and in-sensor processing designs and Magnetic Random Access Memory (MRAM). Section III delineates the proposed PISA architecture, and the supported operations and presents the near-sensor processing-in-DRAM solution. Section IV gives the proposed bottom-up evaluation framework and simulation results. Section V discusses the future work and finally, Section VI concludes this work.

II. BACKGROUND & MOTIVATION

A. Near-Sensor & In-Sensor Processing

Systematic integration of computing and sensor arrays has been widely studied to eliminate off-chip data transmission and reduce ADC bandwidth by combining CMOS image sensor and processors in one chip as known as PNS (2), (13), (21)–(24), or even integrating pixels and computation unit so-called PIS (12), (17), (25)–(28). In (13), photocurrents are transformed into pulse-width modulation signals and a dedicated analog processor is designed to execute feature extraction reducing ADC power consumption. In (2), 3D-stacked column-parallel ADCs and Processing Elements (PE) are implemented to run spatiotemporal image processing. In (29), a CMOS image sensor with dual-mode delta-sigma ADCs is designed to process 1st-conv layer of Binarized-Weight Neural Networks (BWNN). RedEye (30) executes the convolution operation using charge-sharing tunable capacitors. Although this design shows energy reduction compared to a CPU/GPU by sacrificing accuracy, to achieve high accuracy computation, the required energy per frame increases dramatically by 100×. MACSEN (12) as a PIS platform processes the 1st-conv. layer of BWNNs with the correlated double sampling procedure achieving 1000fps speed in computation mode. However, it suffers from humongous area-overhead and power consumption mainly due to the SRAM-based PIS method. In (31), a pulse-domain algorithm uses fundamental building blocks, photodiode arrays, and an ADC to perform near-sensor image processing that reduces design complexity and enhances both cost and speed. Putting all together, there are three main bottlenecks in IoT imaging systems that this work explores and aims to solve: (1) The conversion and storage of pixel values consume most of the power (>96% (12), (32)) in conventional image sensors; (2) the computation imposes a large area-overhead and power consumption in more recent PNS/PIS units and requires extra memory for intermediate data storage; and (3) the system is hardwired so the functionality is limited to simple pre-processing tasks such as 1st-layer BWNN computation and cannot go beyond that.

B. Processing-in-DRAM Platforms

The PIM in the context of main memory (DRAM- [15], [33], [34]) has drawn much attention in recent years mainly due to larger memory capacities and off-chip data transfer reduction as opposed to SRAM-based PIM. Such processing-in-DRAM platforms show significantly higher throughput leveraging multi-row activation methods to perform bulk bit-wise operations by modifying the DRAM cell and/or SA. For example, Ambit (33) uses Triple-Row Activation (TRA) method to implement majority-based AND/OR logic, outperforming Intel Skylake-CPU, NVIDIA GeForce GPU, and even HMC (35) by 44.9×, 32.0×, and 2.4×, respectively. DRISA (36) employs 3T1C- and 1T1C-based computing mechanisms and achieves 7.7× speedup and 15× better energy-efficiency over GPUs to accelerate CNN. However, there are various challenges in such platforms that make them inefficient acceleration solutions. (1) Given \( R = A \circ B \) function (\( \circ \in \{ \text{AND} / \text{OR} \} \)), TRA-based method takes 4 consecutive steps to calculate one result as it relies on row initialization. Therefore TRA method needs an average of 360ns to perform such in-memory operations. Obviously, this row-initialization load could adversely impact the PIM’s energy-efficiency; (2) By simultaneously activating three DRAM cells in TRA method or five cells in (37), (38), the deviation on the Bit-Line is smaller than typical one-cell read operation in DRAM. This can elongate the sense amplification state or even adversely affect the reliability of the result.
Fig. 2: Realizing coarse-grained and fine-grained computation in the proposed hybrid architecture.

C. MRAM as a High-Performance Non-Volatile Memory

With the great advancement of fabrication technology and commercialization of MRAM (e.g., IBM [39] and Everspin [40], [41]), it is becoming a next-generation universal Non-Volatile Memory (NVM) technology, with potential applications in both last-level cache and main memory [42], [43]. Particularly, recent current-induced Spin-Transfer Torque (STT) and Spin-Orbit Torque (SOT)-based MRAMs have greatly changed the state-of-the-art memory hierarchy due to their non-volatility, zero leakage power in un-accessed bit-cell [44], [45], high integration density (2× more than SRAM), high speed (sub-nanosecond) [46], excellent endurance (∼10^{15} cycles [47]), and compatibility with the CMOS fabrication process (back end of the line) [44]. A standard 1-transistor 1-resistor (1T1R) STT-MRAM bit-cell consists of an access transistor and a Magnetic Tunnel Junction (MTJ). A typical MTJ structure consists of two ferromagnetic layers with a tunnel barrier sandwiched between them [48]. One of the layers is a pinned magnetic layer, while the other one is a free magnetic layer. Due to the tunneling magnetoresistance (TMR) effect [48], the resistance of MTJ is high (low) when the magnetization of two ferromagnetic layers are in anti-parallel (parallel). The free layer magnetization could be manipulated by applying a current induced STT [49]. Therefore, it is time for researchers to start in earnest to explore the application of MRAM in new energy-efficient in-memory and in-sensor computing systems that leverage its unique properties.

III. PROPOSED HYBRID PROCESSING-IN-SENSOR/NEAR-SENSOR ARCHITECTURE

Figure 2 shows an overview of the proposed hybrid architecture’s data flow regarding a simple network structure with four convolutional layers and one Fully-Connected (FC) layer. Similarly, our proposed approach can be extended to accelerate much more complex CNN models. We first propose PISA as a flexible, energy-efficient, and high-performance solution for real-time and smart image processing in AI devices. PISA will integrate sensing and processing phases and can intrinsically implement a coarse-grained convolution operation (Fig. 2.4) required in a wide variety of image processing tasks such as classification by processing the 1st-layer in BWNNs. The design will be completed with an on-chip reconfigurable PNS unit to perform a low bit-width coarse-grained convolution on the remaining layers. Once the object is roughly detected at the end of step 2, PISA will switch to typical sensing mode 3 to capture the image for a fine-grained convolution using the near-sensor PIM unit 4.

A. PISA Architecture

1) Compute-Pixel Element: To enable an integrated sensing and processing mode for PISA, we propose to upgrade the conventional pixel unit to a Compute-Pixel (CP). The CP is composed of a pixel (three transistors and one Photodiode (PD)) as shown in Fig. 3.a, and vice compute add-ons. The compute add-on consists of three transistors of which T4 and T5 work as deep triode region current sources and a 2:1 MUX controlled by NVM element. We selected STT-MRAM as the NVM unit as depicted in Fig. 3.a due to its high speed (sub-nanoseconds), long-endurance (10 years), and less than fJ/bit memory write energy (close to SRAM) [50]. Thus, the binary weight data is stored as the magnetization direction in the MTJ’s free layer, which could be programmed through the current-induced STT by NVM write driver. A reference resistor is then used to realize a voltage divider circuit to read out the weight value from the memory. Fig. 4 illustrates a 2×1 CP array implementation. The Ri (Row) signal is controlled by Row Ctrl and shared across CPs located in the same row to enable access during the row-wise sensing mode. However, the CR (ComputeRow) is a unique controlling signal connected to entire CP units activated during processing mode. A Sense Bit-line (SBL) is shared across the pixels on the same column connected to sensor output for sensing-only mode (Fig.
2) Operation Modes: We develop PISA as a high-performance architecture for real-time and smart edge feature extraction as shown in Fig. 5(a) on top of the proposed circuit schemes. At a high level, the PISA array consists of an \( m \times n \) Compute Focal Plane (CFP), row and column controllers (Ctrl), command decoder, sensor timing Ctrl, and sensor I/O operating in two modes, i.e., sensing-only and integrated sensing-processing. The CFP is designed to co-integrate sensing and processing of the \( 1^{st} \)-layer of BWNNS targeting a low-power and coarse-grained detection. The \( 1^{st} \)-layer binarized weight corresponding to each pixel is pre-stored into NVMs and an efficient coarse-grained MAC operation is then accomplished in a voltage-controlled crossbar fashion (Fig. 3(b)). Accordingly, the output of the first layer is transmitted to a PNS or near-sensor PIM-based unit that enables the computation of the remaining BWNNS layers. Once the object is roughly detected at the edge, PISA switches to sensing mode like a traditional rolling-shutter CMOS image sensor. It then transmits raw images to a near sensor unit or an off-chip processor, for a fine-grained bit-wise convulsion operation. Fig. 6(a) depicts a sample FC neural network, wherein CP\(_{1,1}\)-CP\(_{m,n}\) are linked to output via NVM\(_1\)'s weight. Similarly, every pixel is connected to output. To maximize MAC computation throughput and fully leverage PISA’s parallelism, we propose a hardware mapping scheme and connection configuration between CP elements and corresponding NVM add-ons shown in Fig. 6(b) to implement the target neural network. In the following, the two operating modes of PISA are further elaborated.

Sensing Mode: In the sensing mode, by initially setting Rst=’high’, the PD connected to the T1 transistor (see Fig. 3(a)) turns into inverse polarization. In this way, turning on the access transistor T3 and \( k_1 \) switch (see Fig. 3(b)) at the Sensor I/O allows the \( C_1 \) capacitor to fully charge through SBL. By turning off T1, PD generates a photo-current with respect to the external light intensity which in turn leads to a voltage drop (\( V_{PD} \)) at the gate of T2. Once again by turning on the T3 and this time \( k_2 \) switch, \( C_2 \) is selected to record the voltage drop. Therefore, the voltage values before and after the image light exposure, i.e., \( V_1 \) and \( V_2 \), are sampled by the CP, and the difference between two voltages is sensed with an amplifier. This value is proportional to the voltage drop on \( V_{PD} \). In other words, the voltage at the cathode of PD can be read at the pixel output. Please note that in sensing mode, the CR signal is grounded.

Integrated Sensing-Processing Mode: In this mode, as shown in a sample \( 2 \times 1 \) CP array in Fig. 4, the \( C_{PD} \) capacitor is initialized to the fully-charged state by setting Rst=’high’, similar to the sensing mode. During an evaluation cycle, by turning off T1, the row ctrl activates the CR signal, while the \( R_i \) signals are deactivated. This will activate the entire array for a single-cycle MAC operation. The core idea behind compute add-on, shown in Fig. 3(a), is to leverage pixel’s \( V_{PD} \) as a sampling voltage in \( v \)-NVM units to simultaneously generate (pull) current through T4 (T5) on the CBL. To implement multiplications between the pixel value identified by \( V_{PD} \) and the binary weight stored in NVM, a 2:1 MUX unit was devised in every CP taking the T4 and T5 source signals as inputs and NVM sensed data as the selector. Note that T4 and T5 drains are connected to \( V_{DD} \) and \(-V_{DD} \), respectively. After exposure, the set of input sensor voltages \( V_{PD} = [V_{PD_{1,1}}, V_{PD_{1,2}}, ..., V_{PD_{m,n}}] \) is applied to the gate of T4s generating current set \( I_{T4} = \{I_{1,1,1}, I_{1,1,2}, ..., I_{1,1,n}, ..., I_{m,n,1}, ..., I_{m,n,m} \} \) for the entire array. If the binary weight equals ‘1’ (\( W_i=+1 \)), T4 acts a current source and generates a current with \( I_{i,1} \) magnitude on the shared CBL as shown by the red dashed line in Fig. 4. However, if the binary weight equals ‘0’ (\( W_i=-1 \)), T5 transistor acts a negative current source and pulls a current with the same magnitude as \( I_{i,1} \) in the opposite direction from the shared CBL as indicated by the blue dashed line in Fig. 4. This mechanism converts every input pixel value to a weighted current according to the NVM that is interpreted as the multiplication in BWNNS. Mathematically, let \( G_{i,j} \) be the conductance of the synapse connecting \( i^{th} \) to the \( j^{th} \) node, the current through that synapse is \( G_{i,j}V_i \) and the collection of the current through each CBL represents the MAC result

Fig. 4: A \( 2 \times 1 \) CP array in integrated sensing-processing mode.

Fig. 5: (a) The overview of PISA, and (b) PNS architectures.
Fig. 6: (a) An example of fully-connected network with \( v \) output, (b) PISA’s mapping scheme for an \( m \times n \) CFP.

\[
I_{\text{sum},j} = \sum_i G_{ij} V_i,
\]

according to Kirchhoff’s law. This is readily calculated by measuring the voltage across a sensing resistor. For the activation function, we designed and tuned a sense circuit connected to each CBL based on StrongARM latch to realize an in-sensor \( \text{sign} \) function \([51], [52]\) as shown in Fig. 3(c). The SA requires two clock phases: pre-charge (Clk ‘high’) and sensing (Clk ‘low’). During sensing, \( I_{\text{sum}(x)} \) flows from every CBL to the ground and generates a sense voltage \( V_{\text{sense}} \) at the input of the SA. This voltage is compared with the reference voltage by applying a proportional current over a processing reference resistor \( R_{\text{pro}} \) activated by the mode signal. The binary activation is then transmitted through the bus fabrics to the PIM unit for storage.

B. PNS Architecture

Besides 1st-layer, there are other convolutional and FC layers\(^1\) in BWNNs that can be accelerated close to the sensor without sending the activated feature maps to off-chip processors. The general memory organization of the PNS is shown in Fig. 5(b). The memory unit is divided into multiple banks consisting of computational sub-arrays. Every two sub-arrays share a Local Row Buffer (LRB) and the entire array shares a Digital Processing Unit (DPU) to pre-process the data by quantization and post-process outputs with linear batch normalization and activation. We divide the PNS’s sub-array row space into two distinct regions as depicted in Fig. 7(a): 1- Data rows (500 rows out of 512) connected to a regular Row Decoder (RD), and 2- Computation rows (12), connected to a Modified Row Decoder (MRD), which enables two-row activation required for bulk bit-wise in-memory operations between operands.

1) Dual-Row Activation Mechanism: With careful observation of the existing processing-in-DRAM platforms, we realized that they impose reliability concerns and an excessive latency and energy to the memory chip, which could be alleviated by rethinking about SA circuit. Our key idea is to perform an in-memory \text{NAND2} operation as a universal function through a Dual-Row Activation mechanism (DRA) to address these challenges. To achieve this goal, we propose a computational sub-array with new reconfigurable SA, as shown in Fig. 7(a)-(b), developed on top of the existing DRAM circuitry. The new SA consists of a regular DRAM SA with only one add-on inverter with three enable signals \( E_{nM}, E_{nL}, E_{nA} \). This design leverages the charge-sharing feature of the DRAM cell and elevates it to implement \( (N)\text{AND2} \) logic between two selected rows through static capacitive-NAND function in a single cycle. To implement capacitor-based logic, we use an inverter with shifted Voltage Transfer Characteristic (VTC), as shown in Fig. 7(c). In this way, a NAND logic can be readily carried out based on high switching voltage \( V_s \) inverter with standard high-\( V_{th} \) NMOS and low-\( V_{th} \) PMOS transistors. It is worth mentioning that, utilizing low/high-threshold voltage transistors along with normal-threshold transistors has been accomplished in the low-power application, and many circuits have enjoyed this technique in low-power design \([53], [54]\).

To avoid original data overwritten as a common issue in processing-in-DRAM platforms \([33], [55]\), every operand row requires to be initially copied into compute rows before computation. Here, consider \( D_i \) and \( D_j \) operands are copied from data rows to \( x1 \) and \( x2 \) rows and both BL and \( \overline{BL} \) are precharged to \( \frac{V_{dd}}{2} \) (Precharged State in Fig. 8). To implement DRA, the Ctrl first activates two \( WLs \) in computational row space (here, \( x1 \) and \( x2 \)) through the modified decoder for charge-sharing when all the other enable signals are deactivated (Charge Sharing State). During Sense Amplification State, by activating the corresponding enable signals \( E_{nL} \) and \( E_{nA} \), the input voltage of high-\( V_s \) inverter in the reconfigurable SA can be simply derived as \( V_i = \frac{nV_{th}}{C} \), where \( n \) is the number of DRAM cells storing logic ‘1’ and \( C \) represents the total number of unit capacitors connected to the inverter (i.e., 2 in DRA method). Now, the high-\( V_s \) inverter amplifies the deviation from \( \frac{3}{2}V_{dd} \) and realizes a \text{NAND2} function and writes back the inverted result in a single memory cycle.

2) Hardware Mapping: Figure 9 gives an overview of the proposed BWNN bit-wise acceleration steps. In the first step, the preprocessed data from PISA is mapped into the...
PNS’s computational sub-arrays. In the second step, parallel computational sub-arrays, which are designed to handle the computational load employing PIM techniques, perform bulk bit-wise operations between tensors and generate the output. Accordingly, the output is activated by DPU’s active unit and saved back into memory. From a computation perspective, every conv. layer can be similarly implemented by exploiting logic AND, bitcount, and bitshift as rapid and parallelizable operations [5]. Assume \( I \) is a sequence of \( M \)-bit input integers (3-bit as an example in Fig. 9) located in input fmaps covered by sliding kernel of \( W \), such that \( I_i \in I \) is an \( M \)-bit vector representing a fixed-point integer. Now, we index the bits of each \( I_i \) element from LSB to MSB with \( m = [0, M - 1] \), such that \( m = 0 \) and \( m = M - 1 \) are corresponding to LSB and MSB, respectively. Accordingly, we represent a second sequence denoted as \( C_m(I) \) including the combination of \( m^{th} \) bit of all \( I_i \) elements (shown by colored elliptic). For instance, \( C_0(I) \) vector consists of LSBs of all \( I_i \) elements “0110”. Considering \( W \) as a sequence of \( N \)-bit weight integers (3-bit, herein) located in a sliding kernel with index of \( n = [0, N - 1] \). The second sequence can be similarly generated as \( C_n(W) \). Now, by considering the set of all \( m^{th} \) value sequences, the \( I \) can be represented like \( I = \sum_{m=0}^{M-1} 2^m c_m(I) \). Likewise, \( W \) can be represented like \( W = \sum_{n=0}^{N-1} 2^n c_n(W) \). In this way, the convolution between \( I \) and \( W \) can be defined as \( \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} 2^{m+n+1} \text{bitcount}(c_m(I), c_n(W)) \). As shown in the data mapping step of Fig. 9, \( C_2(W) \). \( C_0(W) \) are consequently mapped to the designated sub-array. Accordingly, \( C_2(I) - C_0(I) \) are mapped in the following memory rows in the same way. Now, computational sub-array can perform bit-wise parallel AND operation of \( C_n(W) \) and \( C_m(I) \) as depicted in Fig. 9 leveraging the DRA mechanism. The results stored within the sub-array will be accordingly processed using DPU’s bit-counter. Bit-counter readily adds up the number of “1”s in each resultant vector and passes it to the Shifter unit. As depicted in Fig. 9 “0001”, as result of Bit-Counter is left-shifted by 3-bit (×22+1) to “1000”. Eventually, the PIM adds the shifter unit’s outputs to produce output fmaps for every layer. Note that the PNS supports multi-bit convolution so the various configurations of weight:input can be achieved at the edge.

IV. PERFORMANCE EVALUATION

A. Framework & Methodology

To assess the performance of the proposed design, we developed a simulation framework from scratch consisting of two main components as shown in Fig. 10. First, for coarse-grained computation, at the circuit level, we fully implemented PISA with peripheral circuitry with TSMC 65nm-GP in Cadence to achieve the performance parameters. For the NVM elements, we jointly use the Non-Equilibrium Green’s Function (NEGF) and Landau-Lifshitz-Gilbert (LLG) equations to model MTJ [50]. A Verilog-A model of NVM element is then developed to co-simulate with interface CMOS circuits in Cadence Spectre and SPICE. PISA requires binarizing the \( 1^{st} \)-layer weights as discussed while the rest of the layers processed with the PIM unit have various bit-length. We trained a PyTorch BWNW model inspired by [56, 57] extracting the \( 1^{st} \)-layer weights.
PISA's NVM elements are then programmed at the circuit-level by the binary weights. After 1st-layer computation, the results are recorded and fed into a behavioral-level PIM simulator to simulate the near-sensor PIM platform. Second, for fine-grained computation, at the circuit level, we fully implemented the PNS and DRISA-1T1C [13] with TSMC 65nm-GP in Cadence to achieve the performance parameters. An architecture-level PIM support tool is developed to model the timing, energy, and area based on the circuit-level data. This tool offers the same flexibility in memory configuration regarding bank/mat/subarray organization and peripheral circuitry design as Cacti [58] while supporting PIM-level configurations. Based on the circuit level results, we altered the configuration files (.cflg) with different array organizations and add-ons such as DPU and achieved performance for PIM operations. We then configured the PIM unit with 1024 rows and 256 columns, 4×4 mats per bank organized in an H-tree routing manner, and 16×16 banks (with 1/1 as row/column activation) in each memory group. The behavioral PIM model developed in Python then takes coarse-grained computation results, 2nd-to-last layer trained weights, and the PIM architecture-level data and processes the BWNN. It calculates the latency and energy that the whole system spends executing the network.

**B. Functionality**

Fig. 11 shows the post-layout transient simulation waveforms of a 4×4 PISA array with eight NVM units (i.e., 8 storing binary weights with $V_{\text{CCL}}$, $V_{\text{OUT}}$, $I_{\text{CBL}}$, and $V_{\text{OUT}}$ signals. PISA executes global shutter in processing mode and conducts all computations in parallel. As shown, periodically, by precharging $V_{\text{PD}}$ to VDD, the computation takes place at every falling edge of the clock, i.e., $\sim$100μs. In this way, $I_{\text{CBL}}$ carries the summation current corresponding to $V_{\text{PDS}}$. As can be seen, when $I_{\text{CBL}}$ is positive (e.g., the case of 32μA and 39μA) meaning the MAC result is larger than zero and the output sign function results in “1” and vice-versa. The transient simulation results of the in-DRAM DRA mechanism to realize single-cycle (N)AND2 operation is shown in Fig. 12 for three possible input combinations. We can observe how NAND output and accordingly cell’s capacitor is charged to $V_{\text{dd}}$ (when $D_i,D_j=11$) or discharged to GND (when $D_i,D_j=00/01/10$) during sense amplification state.

**C. Robustness**

We simulated the PISA’s circuit-level variations and noises with equivalent post-layout parasitics at 300K with 10000 Monte-Carlo runs. This includes a variation in width/length of transistors and CBL capacitance. Besides, the impact of thermal noises was modeled as the additive Gaussian noise on the dynamic capacitance along with 1/f noise of CMOS transistors from the source-follower in pixels. Our study shows that percentage of failure upon a considerable variation/noise (10%) across 10000 iterations is 0% as plotted $V_{\text{PDS}}$ in Fig. 11. For variations above 10%, a noise-aware training technique is used injecting multiplicative noise onto the weights in the training to increase BWNN robustness. For the NVM element, we added a $\sigma = 2\%$ variation to the Resistance-Area product, and a $\sigma = 5\%$ process variation (typical MTJ conductance variation [59]) on the TMR and verified a sense margin of 70mV between parallel and anti-parallel cases.

As for PNS unit, we performed a comprehensive circuit-level simulation to study the effect of process variation on both DRA and TRA methods considering different noise sources and variation in all components including DRAM cell (BL/WL capacitance and transistor, shown in Fig. 13) and SA (width/length of transistors-$V_s$). We ran Monte-Carlo simulation (DRAM cell parameters were taken and scaled from Rambus [59] under 10000 trials and increased the amount of variation from ±0% to ±30% for each method. Table I shows the percentage of the test error in each variation. We observe that even considering a significant ±10% variation, the percentage of erroneous DRA across 10000 trials is 0%, where the TRA method shows a failure with 0.18%.

**D. Energy & Performance**

We analyze the PISA's utility in processing the 1st-Conv layer for continuous mobile vision in three scenarios, i.e., assisting mobile CPU (PISA-CPU), assisting mobile GPU (PISA-GPU), and assisting mobile GPU (PISA-GPU).

![Fig. 11: Post-layout transient simulation result for a sample 4×4 PISA array.](image)

![Fig. 12: The transient simulation of a single PNS sub-array. P.S., C.S.S., S.A.S. are short for Precharged State, Charge Sharing State, and Sense Amplification State, respectively.](image)
(PISA-GPU), and PISA-PNS, and compare it with a baseline sensor-CPU platform. For this goal, a BWNN model with 6 binary-weight Conv. layers and 2 FC layers to process the SVHN data-set is adopted. The energy consumption and latency results of the under-test platforms are then reported for four various weight/input configurations in PNS (W:I= 1:32, 1:16, 1:8, 1:4) in Fig. [14]. The under-test platforms in each experiment from left to right include the baseline design consisting of a conventional 128 × 128 image sensor and an Intel(R) Core i7-6700 at 3.4GHz CPU with 16GB RAM where CPU plays the main role in processing all layers after receiving the raw data from the sensor’s ADC. The second platform consists of the same CPU connected to 128 × 128 PISA array, where PISA processes 1st Conv. layer and remaining layers are processed by the CPU. We find that PISA performs differently against conventional CMOS image sensors. First, PISA substantially reduces the data transmission energy by ∼84% paired with the CPU and GPU. The PISA-GPU platform saves 58% energy on average compared with the baseline as shown in Fig. [14a]. While the PISA-GPU does not show a remarkable energy-saving over CPU-GPU but is still 89% more energy-efficient than the baseline. Besides reduction in data transfer, the other reason behind such a striking energy saving is eliminating energy-hungry ADC units in PISA’s processing mode. Second, we observe that PISA-PNS-1 reduces the energy consumption of edge devices dramatically. The PISA-PNS-II requires ∼50-170μJ energy depending on PNS configuration to process the whole BWNN on the edge, which is a safe choice for power-constrained IoT sensor devices. Please note that PISA-PNS designs almost eliminate the data transmission energy. Fig. [14b] illustrates the execution time corresponding to various W:I configurations. We observe that the PISA-PNS-II design achieves ∼3-7× speed-up in processing input frames compared with the baseline. However, PISA-PNS-I indicates a shorter execution time.

**E. Resource Utilization**

To explore the impact of PISA in reducing memory bottleneck in executing the 1st-layer of BWNN, we measured the time fraction at which on/off-chip data transfer limits the performance as shown in Fig. [15a]. This evaluation was accomplished through experimentally extracted results of each platform with the number of memory access. We observe the PISA-PNS platforms spend less than 22% of time for data conversion and memory access, whereas the baseline design spends over 90% of its time waiting to load data from memory. A low memory bottleneck ratio can be translated to a high resource utilization ratio as depicted in Fig. [15b]. We observe that PISA-PNS platforms obtain the highest ratio utilizing up to 83% computation resources.

![Figure 13: Noise sources in DRAM cell. Glossary: Cubl, Cs, and Ccross are WL- BL, BL-substrate, and BL-BL capacitance, respectively.](image1)

![Figure 14: (a) Energy consumption, and (b) Execution time of under-test PISA-based platforms in various configurations compared with the baseline.](image2)

![Figure 15: (a) Memory bottleneck ratio, (b) Resource Utilization ratio.](image3)
TABLE II: Performance comparison of various PIS units.

| Designs | Technology (nm) | Purpose | Comput. Scheme | Memory | NV* Pixel Size (µm²) | Array Size | Frame Rate (frame/s) | Power (µW) | Efficiency (TOP/s/W) |
|---------|-----------------|---------|----------------|--------|----------------------|-----------|----------------------|-----------|---------------------|
| MNIST  | 180             | 2D optic low est. | raw-wise       | No     | 28.8×28.8            | 64×64     | 50                   | 0.0029    | 0.0041             |
| SVHN   | 180             | edge/blur/sharpen/1st layer DNN | raw-wise | No     | 7.6×7.6              | 128×128   | 480                  | sensing: 0.077/0.091 | 0.777     |
| CIFAR-10 | 60/90         | STP†    | raw-wise       | Yes    | 3.5×3.5              | 1296×976  | 1000                 | sensing: 2.30/363 | 0.386     |
| PISA   | 180             | 1st layer BNN   | entire-array   | No     | 110×110              | 32×32     | 1000                 | 0.0124    | 1.32               |
|        | 180             | edge/blur/sharpen/1st layer DNN | raw-wise | Yes    | 32.6×32.6            | 256×256   | 100,000              | 1.23      | 0.535              |

Table II compares the structural and performance parameters of selective PIS designs in the literature. As different designs are developed for specific domains, for an impartial comparison, we estimated and normalized the power consumption when all PIS units execute the similar task of processing the 1st-layer of DNN. The PISA achieves the frame rate of 1000 and the efficiency of ~1.745 TOP/s/W as the most efficient design. This comes from the massively-parallel CFP and eliminating ADC for coarse-grained detection. However, the design in [11] achieves the highest frame-rate and the design in [2] imposes the least pixel size enabling in-sensor computing. As for the area, our post-layout simulation results reported in Table I show a PISA’s compute-pixel occupies ~55×55 µm² in 65nm. As we do not have access to the other layouts’ configurations, it is very hard to have a fair comparison between area overheads. However, we believe a ballpark assessment can be made by comparing the number of minimum size transistors in previous SRAM-based designs and PISA’s lower-overhead compute add-on. We reimplemented MACSen [12] at circuit-level as the only BWNN accelerator developed with the same purpose. Our evaluation showed that with the same near-sensor unit based on DRISA [15], PISA consumes ~40% less power consumption. Putting everything together, PISA offers 1) a low-overhead, dual-mode and reconfigurable design to keep the sensing performance and realize a processing mode to remarkably reduce the power consumption of data conversion and transmission; 2) single-cycle in-sensor processing mechanism to improve image processing speed; 3) highly parallel in-sensor processing design to achieve ultra-high-throughput; 4) exploiting NVM which reduces standby power consumption during idle time and offers instant wake-up time, and resilience to power failure to achieve high performance.

**G. Accuracy**

In the original BWNN topology, all the layers, except the first and last, are implemented with binarized weights [52], [60], [61]. Since, in image classification tasks, the number of input channels is relatively smaller than the number of internal layers’ channels, required parameters and computations are small. Thus, converting the input layer will not be a significant issue [52]. We conduct experiments on several datasets, including MNIST, SVHN, and CIFAR-10. A BWNN model with 6 binary-weight Conv. layers and 2 FC layers to process the SVHN data-set is adopted. The 1st-layer consists of 32-by-32 images centered around a single character, where each pixel is mapped into a CP unit in PISA, the output states of PISA are then fed into the second layer implemented by near-memory design. Fig. 16 shows the validation error versus the number of epochs of three different datasets in a worst-case scenario, i.e., with 1:4 configuration for 2nd to the last layer. The comparison of classification accuracy is summarized in Table III. We find that the PISA shows an acceptable accuracy while providing significant energy-delay-product reduction as discussed earlier.

TABLE III: BNN accuracy (%) on MNIST, SVHN and CIFAR-10.

| Configuration | MNIST | SVHN | CIFAR-10 |
|---------------|-------|------|----------|
| [52]          | 96.0  | 97.47| 89.85    |
| [60]          | 98.25 | 97.00| 86.98    |
| [61]          | 98.4  | 94.9 | 80.1     |
| Ours          | 96.9  | 93.05| 88.61    |
|               | 95.12 | 90.35| 79.80    |

V. DISCUSSION AND FUTURE WORK

Although almost all the state-of-the-art image sensor designs utilize effective methods to reduce dynamic energy consumption, including clock gating and low-voltage operation, an increasing number of modern intelligent sensors and more application scenarios, making the standby power dissipation of such systems a critical issue, which can limit the wider sensors’ applications. The emergence of energy harvesting systems as a promising approach for battery-less IoTs suffers from intermittent behavior, leading to data and environmental inconsistencies. For example, captured data by sensors become unstable if they are held for a long time without intermittent resilient architectures and/or harvestable sources. Moreover, since concurrency with sensors is relatively interrupt-driven, intermittency makes this concurrency control much more complex. To solve the data consistency, PISA utilizes NVM
elements, which reduces standby power consumption during idle time, instant wake-up time, and resilience to power failure, leading to high throughput and high performance at the cost of the minor accuracy degradation. Due to the page limit, we plan to extend our future work to investigate image sensors’ challenges in the presence of power failure for energy harvested systems, and more thoroughly discuss PISA’s power failure resiliency.

VI. CONCLUSION

In summary, this work proposed an efficient processing-in-sensor accelerator, namely PISA, for real-time edge-AI devices. PISA intrinsically performs a coarse-grained convolution operation on the 1st-layer of binarized-weight neural networks leveraging a novel compute-pixel with non-volatile weight storage. The design was then completed by a near sensor processing-in-DRAM unit to perform a fine-grained convolution operation over the remaining layers. Our results demonstrate acceptable accuracy on various data sets, where PISA achieves the frame rate of 1000 and the efficiency of ~1.74 TOPs/W.

REFERENCES

[1] T.-H. Hsu, Y.-C. Chiu, W.-C. Wei, Y.-C. Lo, C.-C. Lo, R.-S. Liu, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, “AI edge devices using computing-in-memory and processing-in-sensor: from system to device,” in 2019 IEEE International Electron Devices Meeting (IEDM). IEEE, 2019, pp. 22–5.

[2] T. Yamazaki, H. Katayama, S. Uehara, A. Nose, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, Y. Hisamatsu, S. Matsumoto, L. Miyashita, Y. Watanabe, T. Izawa, Y. Muramatsu, and M. Ishikawa, “4.9 a 1ms high-speed vision chip with 3d-stacked 140gops column-parallel pes for spatio-temporal image processing,” in 2017 IEEE International Solid-State-Circuits Conference (ISSCC). IEEE, 2017, pp. 82–83.

[3] M. Gottardi and M. Lecca, “A 64 × 64 pixel vision sensor for local binary pattern computation,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 5, pp. 1831–1839, 2018.

[4] J. H. Ko, M. F. Amir, K. Z. Ahmed, T. Na, and S. Mukhopadhyay, “A single-chip image sensor node with energy harvesting from a cmos pixel array,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 9, pp. 2295–2307, 2017.

[5] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” arXiv preprint arXiv:1606.06160, 2016.

[6] A. Haj-Ali, R. Ben-Hur, N. Wald, R. Ronen, and S. Kvatinsky, “Imaging: In-memory algorithms for image processing.” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 12, pp. 4258–4271, 2018.

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Adrian: Efficient processing-in-sensor architecture integrating mac operations into image sensor for ultra-low-power bnn-based intelligent visual perception,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 2, pp. 627–631, 2020.

[8] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, “Drisa: A dram-based reconfigurable in-situ accelerator,” in Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2017, pp. 298–301.

[9] Y. Zhou, H. Cai, L. Xie, M. Han, M. Liu, S. Xu, B. Liu, W. Zhao, and J. Yang, “A self-timed voltage-mode sensing scheme with successive sensing and checking for stt-mram,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 5, pp. 1602–1614, 2020.

[10] T.-H. Hsu, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, “A 0.5v real-time computational cmos image sensor with programmable kernel for feature extraction,” IEEE Journal of Solid-State Circuits, vol. 56, no. 5, pp. 1588–1596, 2020.

[11] Z. Liu, E. Ren, F. Qiao, Q. Wei, X. Liu, L. Luo, H. Zhao, and H. Yang, “Ns-cim: A current-mode computation-in-memory architecture enabling near-sensor processing for intelligent 12-nm nodes,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 2909–2922, 2020.

[12] S. Li, D. Niu, T. K. Malladi, H. Zheng, B. Brennan, and Y. Xie, “Drissa: A dram-based reconfigurable in-situ accelerator,” in Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2017, pp. 298–301.

[13] T.-H. Hsu, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, “A 0.5v real-time computational cmos image sensor with programmable kernel for feature extraction,” IEEE Journal of Solid-State Circuits, vol. 56, no. 5, pp. 1588–1596, 2020.
[32] J. Choi, S. Park, J. Cho, and E. Yoon, “An energy/illumination-adaptive cmos image sensor with reconfigurable modes of operations,” IEEE Journal of Solid-State Circuits, vol. 50, no. 6, pp. 1438–1450, 2015.

[33] S. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. E. Kozuch, O. Mattu, P. B. Gibbons, and T. C. Mowry, “Ambic: In-memory accelerator for bulk bitwise operations using commodity dram technology,” in 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2017, pp. 273–287.

[34] G. Dai, T. Huang, Y. Chi, J. Zhao, G. Sun, Y. Liu, Y. Wang, X. Xie, and H. Yang, “Graphph: A processing-in-memory architecture for large-scale graph processing,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 4, pp. 640–653, 2018.

[35] “Hybrid memory cube speciﬁcation 2.0.” [Online]. Available: http://www.hybridmemorycube.org/files/SiteDownloads/HMC-30G-VSR_HMCC_Specification_Rev2.0_Public.pdf.

[36] A. Driskill-Smith, D. Apalkov, V. Nikitin, X. Tang, S. Watts, D. Lottis, J. Choi, S. Park, J. Cho, and E. Yoon, “An energy/illumination-adaptive on-chip memory for parallel dnn inferences,” in 2011 3rd IEEE International Conference on Simulation of Semiconductor Processes and Devices (ICSPPD), 2011, pp. 1–3.

[37] M. F. Ali, A. Jaiswal, and K. Roy, “In-memory low-cost bit-serial addition using commodity dram technology,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 1, pp. 155–165, 2019.

[38] S. Angizi and F. Fan. “Graphphie: A graph processing accelerator leveraging in-dram-computing,” in Proceedings of the 2019 on Great Lakes Symposium on VLSI, 2019, pp. 45–50.

[39] W. J. Gallagher and S. S. Parkin, “Development of the magnetic tunnel junction mram at ibm: From first junctions to a 16-mb mram demonstrator chip,” IBM Journal of Research and Development, vol. 50, no. 1, pp. 5–23, 2006.

[40] “Everspin announces sampling of the world’s first 1-gigabit mram product.” [Online]. Available: https://www.everspin.com.

[41] “Emsd4e01g - 1gb spin-transfer torque mram” (2020). [Online]. Available: https://www.everspin.com/family/emsd4e01g/.

[42] Y. Hui, “Spin-transfer torque mram (stt-mram): Challenges and prospects,” AAPPs bulletin, vol. 18, no. 6, pp. 33–40, 2008.

[43] W. Kang, Y. Ran, Y. Zhang, W. Lv, and W. Zhao, “Modeling and exploration of the voltage-controlled magnetic anisotropy effect for the next-generation low-power and high-speed mram applications,” IEEE Transactions on Nanotechnology, vol. 16, no. 3, pp. 387–395, 2017.

[44] S. Fukami, T. Anekawa, C. Zhang, and H. Ohno, “A spin-orbit torque switching scheme with collinear magnetic easy axis and current conﬁguration,” Nature nanotechnology, 2016.

[45] H. Zhao, B. Glass, P. K. Amiri, A. Lyle, Y. Zhang, Y.-J. Chen, G. Rowlands, P. Upadhyaya, Z. Zeng, J. A. Katine, J. Langer, K. Galatsis, H. Jiang, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, “Sub-200 ps spin transfer torque switching in in-plane magnetic tunnel junctions with interface perpendicular anisotropy,” Journal of Physics D: Applied Physics, vol. 45, no. 2, p. 025001, 2011.

[46] G. E. Rowlands, T. Rahman, J. A. Katine, J. Langer, A. Lyle, H. Zhao, J. G. Alzate, A. A. Kovalyev, Y. Tserkovnyak, Z. M. Zeng, H. W. Jiang, K. Galatsis, Y. M. Hui, P. K. Amiri, K. L. Wang, I. N. Krivorotov, and J.-P. Wang, “Deep subnanosecond spin torque switching in magnetic tunnel junctions with combined in-plane and perpendicular polarizers,” Applied Physics Letters, vol. 98, no. 10, p. 102509, 2011.

[47] J. J. Kan, C. Park, C. Ching, J. Ahn, L. Xue, R. Wang, A. Kontos, S. Liang, M. Bangar, H. Chen, S. Hassan, S. Kim, M. Pakala, and S. H. Kang, “Systematic validation of 2x nm diameter perpendicular mtj arrays and mgo barrier for sub-10 nm embedded stt-mram with practically unlimited endurance,” in Electron Devices Meeting (IEDM), 2016 IEEE International. IEEE, 2016, pp. 27–4.

[48] T. Kawahara, “Challenges toward gigabit-scale spin-transfer torque random access memory and beyond for normally off, green information technology infrastructure,” Journal of Applied Physics, vol. 109, no. 7, p. 071325, 2011.

[49] M. D. Stiles and A. Zangwill, “Anatomy of spin-transfer torque.” Physical Review B, vol. 66, no. 1, p. 014407, 2002.

[50] X. Fong, S. K. Gupta, N. M. Mojumder, S. H. Choday, C. Augustine, and K. Roy, “Knack: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque mram bit-cells,” in 2011 International Conference on Simulation of Semiconductor Processes and Devices, 2011, pp. 51–54.

[51] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” arXiv preprint arXiv:1511.00932 2015.