RoboVisio: A Micro-Robot Vision Domain-Specific SoC for Autonomous Navigation Enabling Fully-on-Chip Intelligence via 2-MB eMRAM

Qirui Zhang, *Member, IEEE*, Zichen Fan, *Graduate Student Member, IEEE*, Hyochan An, Zhehong Wang, Ziyun Li, *Member, IEEE*, Guanru Wang, Pierre Abillama, Hun-Seok Kim, *Senior Member, IEEE*, David Blaauw, *Fellow, IEEE*, and Dennis Sylvester, *Fellow, IEEE*

***Abstract*— This article presents RoboVisio, an efficient and highly flexible domain-specific system-on-chip (SoC) for vision tasks in fully autonomous micro-robot navigation. A novel hybrid processing element (PE) is proposed, in which classic vision tasks achieve high efficiency by using a *2-D-mapping* architecture, while convolutional neural network (CNN) is executed in an efficient *output-channel-parallel* systolic manner. Combining both processing schemes into a single PE array future-proofs the architecture, facilitating next-generation CNN-heavy vision algorithms, while saving 40% area and leakage with no power overhead and throughput loss, compared with two separate array implementations. To further improve energy and area efficiency, the design incorporates a number of key features: 1) 2-MB magnetoresistive random access memory (MRAM) for non-volatile fully-on-chip weight storage; 2) a unified image-activation memory (IAMEM) with block-swapping-based input/output image buffering that reduces buffer footprint by 50% and eliminates data copy for multi-frame buffering; and 3) a combination of weight buffering and CNN loop ordering that reduces weight memory system power by 75%. Fabricated in 22-nm CMOS, the design achieves 0.22 nJ/pixels for Harris corner feature detection (a classic or non-CNN vision task) and 3.5 TOPS/W (16-bit OP) for CNN, a 40%–170% efficiency improvement over state-of-the-art edge machine learning (ML) SoCs using non-volatile memory (NVM).**

***Index Terms*— Autonomous navigation, computer vision, domain-specific accelerator (DSA), micro-robot, neural network (NN), non-volatile memory (NVM).**

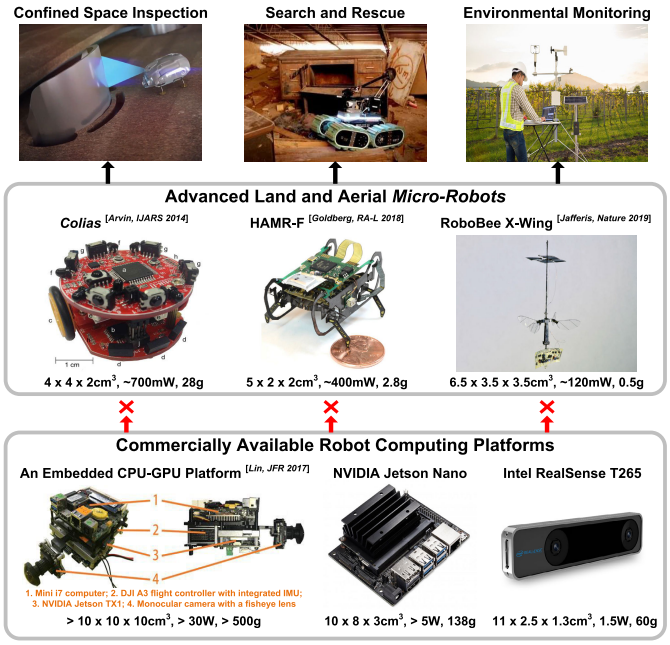


Fig. 1. Applications and computing challenges for advanced micro-robots.

I. Introduction

MOBILE land and aerial micro-robots play a vital role in various critical applications where human intervention is risky or impractical, such as confined space inspection, search and rescue, and environmental monitoring. Notable micro-robot designs include Colias [1], HAMR-F [2], and RoboBee X-Wing [3]. As shown in Fig. 1, they are characterized by their small sizes (centimeter scale), light weights (∼10 g), and low-power consumption (<1 W). Autonomy is essential for these micro-robots, which relies on sophisticated navigation systems, such as those proposed in [4] and [5]. They typically function based on camera and inertial sensor data and consume substantial computing resources. Many commercial robot computing platforms have been developed, such as NVIDIA Jetson Nano and Intel RealSense T265. However, their form factors, weights, and power consumption greatly exceed the constraints for micro-robots and cannot be re-purposed for micro-robot systems. Furthermore, the existing micro-robots, such as HAMR-F [2], already allocate most of their power to mechanical actuation, leaving limited power budget for higher level autonomy. To address this, the extra

Manuscript received 2 October 2023; revised 5 January 2024 and 9 February 2024; accepted 17 February 2024. Date of publication 27 February 2024; date of current version 25 July 2024. This article was approved by Associate Editor Ben Keller. This work was supported by Arm. *(Corresponding author: Qirui Zhang.)*

Qirui Zhang, Zichen Fan, Pierre Abillama, Hun-Seok Kim, David Blaauw, and Dennis Sylvester are with the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: qiruizh@umich.edu).

Hyochan An is with Apple Inc., Cupertino, CA 95014 USA.

Zhehong Wang and Ziyun Li are with Meta Platforms Inc., Menlo Park, CA 94025 USA.

Guanru Wang is with Cisco Systems Inc., San Jose, CA 95134 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSSC.2024.3368350.

Digital Object Identifier 10.1109/JSSC.2024.3368350

0018-9200 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

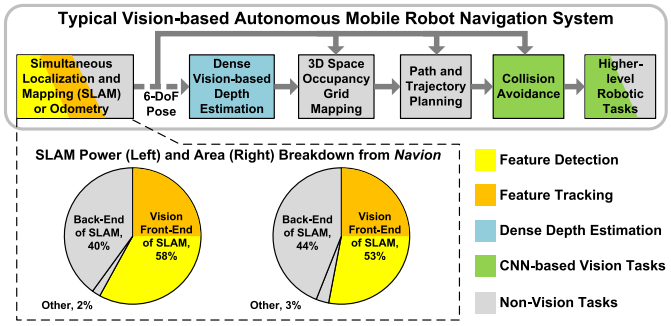


Fig. 2. Vision-based autonomous navigation system for mobile robots.

computing power needed for autonomy should incur negligible overhead, ideally less than 10% of the system's power. Following that constraint, the ∼120-mW total power consumption of RoboBee X-Wing [3], one of the most advanced aerial micro-robots, allows only ∼10 mW to be consumed by the extra computing. In addition, computing hardware for micro-robots must be highly integrated and area-efficient to match their small form factors and weights. Thus, optimizing energy and area efficiency presents significant challenges for micro-robot computing.

To tackle these challenges, a comprehensive study of autonomous navigation systems is essential. Fig. 2 illustrates a typical vision-based autonomous navigation system for mobile robots, based on [5]. The first crucial step is simultaneous localization and mapping (SLAM) [6], [7], which provides six degree of freedom (6-DoF) poses (translation and rotation) for the robot. SLAM comprises a vision front end and an optimization back end. The front end establishes visual correspondences between views of the same landmarks across consecutive frames using feature detection methods, such as Harris corner [8] or Shi–Tomasi [9], and feature tracking methods, such as Lucas–Kanade optical flow (LK flow) [10], [11]. The back end employs non-linear optimizations, mainly involving matrix computations, to determine poses that align with observed visual correspondences based on geometric principles. The next step involves estimating dense depths, which can be achieved using classic methods, such as stereo local matching [12] or convolutional neural network (CNN) [13]. With dense depth information, an accurate and high-resolution occupancy grid map [14] is constructed, facilitating path and trajectory planning. During path traversal, real-time obstacle detection and avoidance are crucial, with CNN-based methods, such as DroNet [15] gaining popularity. Higher level tasks, such as object detection and recognition, are typically CNN-based as well. Among these tasks, vision tasks constitute a significant portion in terms of both quantity and computing resources. As depicted in Fig. 2, the vision front end is a major contributor to power consumption (58%) and area usage (53%) in the state-of-the-art SLAM accelerator, *Navion* [16]. Efficiently accelerating vision tasks has emerged as a prominent challenge in micro-robot computing.

To that end, many computer vision processors have been proposed. Most prior works have focused on dedicated accelerators for specific tasks, such as feature detection [17], optical flow [18], dense depth estimation [19], [20], SLAM [21], [22], or custom algorithms [23]. While these accelerators can be highly efficient, integrating multiple of them into a micro-robot vision system-on-chip (SoC) results in high non-recurring engineering (NRE) costs, and substantial area and leakage overhead. Furthermore, without programmability, advancements in algorithms can quickly render these accelerators obsolete, leading to recursive NRE costs. On the other hand, programmable domain-specific SoCs for micro-robot vision offer forward compatibility and reduced NRE costs. Some designs [24], [25] integrate dedicated accelerators with general-purpose processors to enhance programmability. However, efficiency may be compromised when new vision tasks have to be mapped onto general purpose processors. Image processing domain-specific accelerators (DSAs) [26], [27] provide flexibility and efficiency for classic vision tasks heavily involving image filtering, but may suffer inefficiency for CNN-based tasks, which often require different data flows. Recently, several neural network (NN) DSAs [28], [29], [30], [31], [32], [33] have been proposed for efficient NN inference, but lack flexibility in efficiently supporting classic vision tasks, which remain crucial. Therefore, the need has arisen for micro-robot vision architectures capable of handling both classic and CNN-based vision tasks in an efficient and programmable manner.

In this article, we introduce RoboVisio, an efficient and highly flexible micro-robot vision domain-specific SoC designed for fully autonomous navigation. RoboVisio focuses on the four primary vision tasks that are sequentially executed in a navigation vision pipeline: feature detection, feature tracking, dense depth estimation, and CNN-based obstacle avoidance. To accelerate them efficiently, we propose a novel neural vision processing unit (NVPU) with a hybrid processing element (PE). Classic vision tasks are optimized using a *2-D-mapping* architecture, while CNN processing adopts an efficient *output-channel-parallel* systolic approach. Combining both processing schemes into a single PE array ensures future compatibility with CNN-heavy vision algorithms, while saving 40% area and leakage with no power overhead and throughput loss, compared with two separate array implementations. To further improve energy efficiency, RoboVisio incorporates a number of key features: 1) 2-MB magnetoresistive random access memory (MRAM) for non-volatile fully-on-chip weight storage; 2) a unified image-activation memory (IAMEM) with block-swapping-based input/output image buffering that reduces buffer footprint by 50% and eliminates data copy for multi-frame buffering; and 3) a combination of weight buffering and CNN loop ordering that reduces measured weight memory system power by 75%. Fabricated in 22-nm CMOS, the design achieves 0.22 nJ/pixels for Harris corner feature detection (a classic vision task) and 3.5 TOPS/W (16-bit OP) for CNN, a 40%–170% efficiency improvement over state-of-the-art non-volatile memory (NVM)-based edge machine learning (ML) SoCs.

The rest of this article is organized as follows. Section II provides an overview of the proposed SoC's architecture. Section III discusses data flows, PE design, and instruction set

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

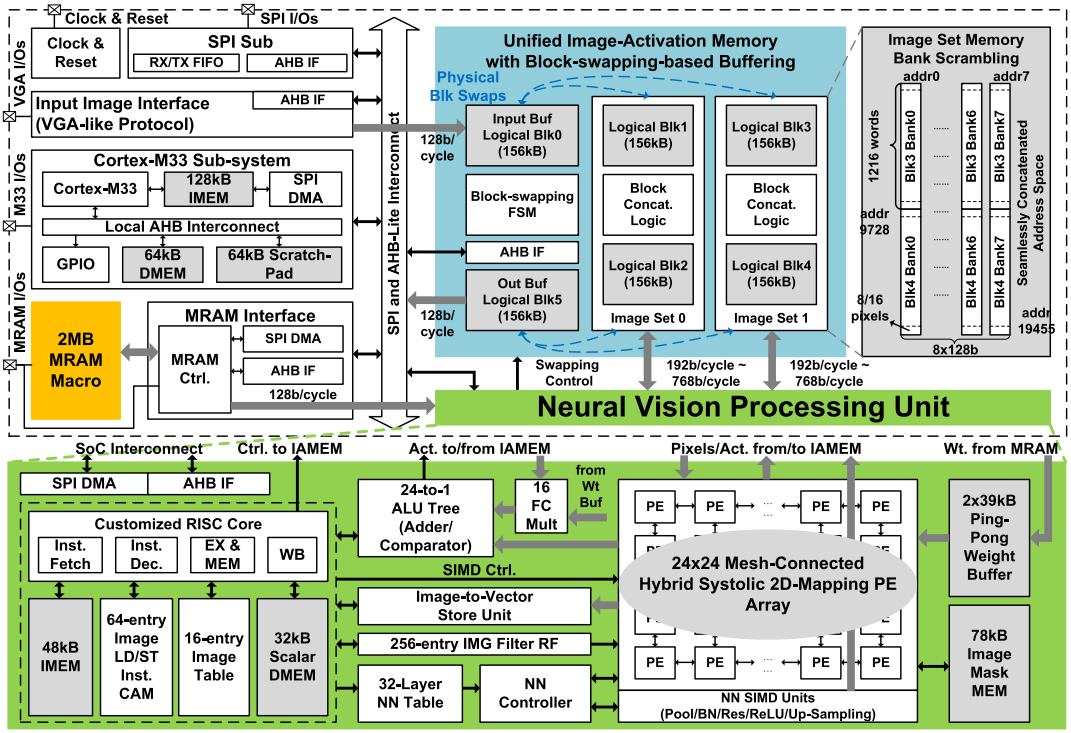


Fig. 3. SoC architecture of RoboVisio.

architecture (ISA) for the NVPU. Section IV explores on-chip memory system optimizations, including image buffering and weight buffering. Section V presents the measurement results of RoboVisio. Finally, Section VI concludes this article.

II. System-on-Chip Architecture

Fig. 3 presents an overview of the proposed SoC's top-level architecture. It features an ARM Cortex-M33 sub-system serving as the central controller, equipped with 128-kB instruction memory, 64-kB data memory, 64-kB scratchpad memory, and floating-point units (FPUs). The on-chip components are integrated using ARM's Advanced Microcontroller Bus Architecture (AMBA) Advanced High-performance Bus (AHB) interconnect. For external communication, the SoC employs a serial peripheral interface (SPI) for program loading, debugging, and external control. Real-time image frames are supplied to the IAMEM through an image interface that utilizes a video graphics array (VGA)-like pixel-serial bit-parallel protocol, supporting up to 20 frames/s at a resolution of 640 × 480 (VGA). The NVPU efficiently handles both classic and CNN-based vision tasks and communicates with the Cortex-M33 core through interruption-based handshaking, particularly for tasks requiring floating-point arithmetic, such as LK flow. The 936-kB IAMEM stores images in either 8- or 16-bit pixels through packing 16 or 8 pixels in the 128-bit SRAM word. It supports simultaneous input and output buffering as well as access to the two scratchpads (image sets), providing a bandwidth of 128 bit per cycle for buffering and up to 768 bit per cycle for image sets. The simultaneous access to image sets allows the NVPU to utilize ping-pong processing schemes and minimize memory latency. Furthermore, the SoC integrates 2-MB MRAM for retentive, high-density, and fully-on-chip weight storage, offering a bandwidth of up to 128 bit per cycle. The MRAM interface supports multi-cycle read to accommodate processing logic of higher frequencies.

III. Neural Vision Processing Unit

A. Architecture Support for Vision Processing Kernels

The design of a programmable DSA begins with the extraction of common processing kernels for the various workloads covered in a target domain. For the micro-robot navigation vision pipeline targeted by RoboVisio, four primary processing kernels are identified: filtering of images, pixelwise operations, reduction of pixel data, and convolution of activations. The first three are used primarily for classic vision tasks, while the last one is for CNN-based vision. These kernels operate on 2-D data structures, and 2-D-mesh connected architectures have been efficient choices [22], [34] for such tasks. We discovered that two mesh architectures, respectively, optimized for image filtering and CNN share many similarities in topology and data flow, which can be unified into one for flexibility and area efficiency without compromising energy efficiency: the *2-D-mapping* architecture [35] optimized for image filtering and the *output-channel-parallel* systolic convolution architecture [36] suited for CNN. This results in a hybrid PE array with two modes: image processing mode (IPM) for classic vision tasks using the *2-D-mapping* data flow and systolic convolution mode (SCM) for CNN-based vision tasks employing the *output-channel-parallel* systolic convolution data flow. Fig. 3 (bottom) illustrates the NVPU architecture, which includes a custom reduced instruction set computer (RISC) core for control and scalar operations, making NVPU instruction programmable. The hybrid PE array is tightly coupled to the RISC core, forming a single instruction, multiple

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

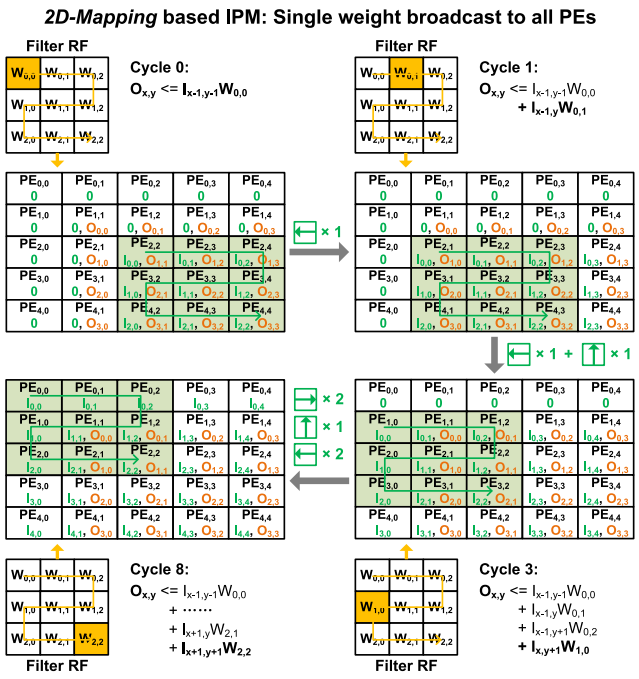


Fig. 4. IPM architecture and data flow for image filtering.

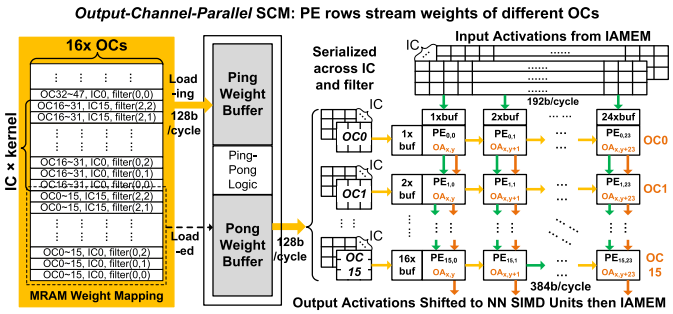


Fig. 5. SCM architecture and data flow for CNN inference.

data (SIMD) datapath. Further details about the NVPU's ISA will be provided in Section III-C. Next, we discuss how NVPU architecturally supports the vision processing kernels.

Filtering of Images: Fig. 4 depicts the *2-D-mapping* data flow employed by IPM for image filtering, using 3×3 Sobel filtering as an example. In IPM, output pixels remain stationary in the PEs, while input pixels can be shifted around. To filter an image block, pixels are initially loaded from IAMEM into shift registers (Shift-Reg) in the PEs, one row per cycle. Then, input pixels start to shift in a zig-zag pattern, and the access of filter weight shifts accordingly, with a single weight broadcast to all PEs. During each shifting cycle, the input pixel in a PE is multiplied by the broadcast weight, and the partial result is accumulated into the output pixel in the same PE. After shifting (nine cycles for example), the output pixels are shifted out of the PE array and stored to IAMEM row by row. Alternatively, they can remain in the PE array for potential reuse in later operations, thereby amortizing IAMEM access overhead.

Pixelwise Operations: IPM also efficiently supports pixelwise operations through associating each PE with a pixel and executing SIMD operations. As will be discussed in Section III-B, each PE is equipped with a multi-function arithmetic logic unit (ALU) and a local register file (LRF). These features facilitate the flexible execution of diverse pixelwise operations essential for vision processing tasks.

Reduction of Pixel Data: Classic vision tasks often involve reduction operations on pixel data. For instance, Harris corner feature detection requires finding the global maximum among Harris corner scores (one per pixel). Similarly, sparse LK flow involves summing processed pixels in an image window. As the 2-D-mesh topology is not well suited for efficient reduction, we have designed an ALU tree closely integrated with the PE array, operating directly on data shifted out of the array. It is pipelined to match the critical path in the PE array, and for an image block, it incurs no more latency than IAMEM load or store. This ALU tree supports common reduction operations, such as addition for summation and comparison for locating minimum/maximum values, with results stored in the RISC core RF.

Convolution of Activations: To efficiently process CNN inference, the hybrid PE array switches to SCM, as Fig. 5 shows. In SCM, CNN activations and weights are stored in 8 bit. SCM shares many similarities with IPM, such as the 2-D-mesh topology, row-by-row IAMEM access, stationary output, and multiply accumulate (MAC) as the primary operation. Unlike IPM, in SCM, each row of PEs maps weights from different output channels. Weights (from ping-pong weight buffers) and input activations (from IAMEM) are serialized across input channels and filter windows and streamed into the PE array in a systolic fashion (non-zig-zag). Due to its systolic nature, weights can be reused across a row of PEs, and activations can be reused across a column of PEs. This reduces memory access to array boundaries, matching the memory bandwidth of IPM. In contrast to the IAMEM load and store latency overhead induced by row-by-row access in IPM, SCM convolution is fully pipelined and incurs no memory latency once set up. When the last inputs of an inner loop iteration reach the bottom-right PE, all output activations for that iteration are ready and can be shifted out of the PE array via IPM's Shift-Reg network. They are either sent to the NN SIMD units (e.g., batch normalization) for layer fusion or stored to IAMEM. Simultaneously, inputs for the next inner loop iteration have already begun streaming. Since convolution layers typically have many input channels (e.g., >32), the output shifting latency of the current iteration can often be hidden beneath the input streaming of the next iteration, enabling continuous input streaming.

B. Hybrid Systolic 2-D-Mapping Processing Element

The PE array architectures of IPM and SCM share highly similar inter-PE connections and the use of MAC units. Therefore, it is advantageous to unify their PE microarchitectures into one design for improved area efficiency and reduced leakage. Fig. 6 shows the proposed hybrid systolic *2-D-mapping* PE. For the *2-D-mapping* PE and systolic convolution PE, their MAC units and accumulation registers (Acc-Reg) are

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

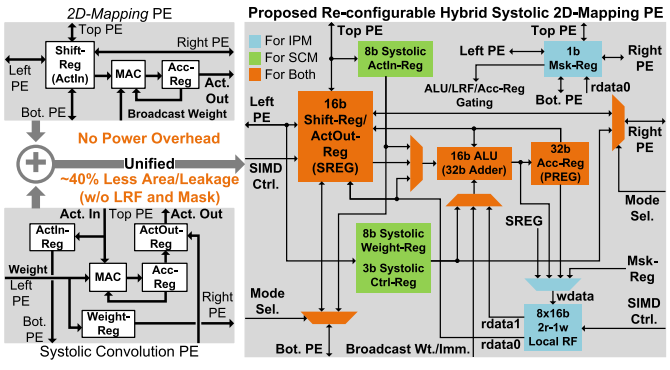


Fig. 6. Hybrid systolic *2-D-mapping* PE microarchitecture.

merged. In addition, the output activation register (ActOut-Reg) in SCM PE is combined with the Shift-Reg in IPM PE. Table I presents the post-synthesis power, performance, and area (PPA) results (22-nm CMOS at 0.8 V and 10 MHz) for various PE designs. The highest frequencies are achieved by pushing the synthesis clock period constraints at 0.8 V. By merging the Shift-Reg with ActOut-Reg and reusing MAC and Acc-Reg, area and leakage of the proposed design are reduced by ∼40% with no power overhead compared with the brute-force design using two separate PEs.

While quantization down to 4 bit can still deliver satisfactory accuracy for NN [37], [38], classic vision tasks may demand significantly larger dynamic ranges. In [27], 16- and 32-bit ALUs are employed to support dense LK flow, mainly due to the requirement from computing products of 8-bit pixel-like variables (pixels, image difference, image gradient, and so on). As our hybrid PE caters to both CNN-based and classic vision tasks, we have opted for 16-bit MAC and 32-bit accumulator to mainly address the requirements of classic vision tasks. In SCM, the 8-bit activations and weights of CNN are sign-extended to 16 bit for computation.

To enhance the flexibility of the hybrid PE for pixel-wise operations, the MAC unit is extended to function as a programmable ALU supporting operations, such as MAC, addition, multiplication, shift, and comparison. Each PE is further equipped with an LRF, enabling intermediate results to remain in the PE array for reusing, which significantly reduces IAMEM access and footprint. For example, Harris corner feature detection requires multiple rounds of filters and pixelwise operations, for which IAMEM access and footprint can be reduced by at least 14× and 3×, respectively, with reusing.

Furthermore, by introducing the mask register (Msk-Reg), NVPU supports the masking mechanism in SIMD execution. For a masked PE (Msk-Reg is 1), only SREG and Msk-Reg are active; all its other units are gated. Masking primarily serves the purpose of reducing power. For example, in 3 × 3 filtering (Fig. 4), two rows/columns of boundary PEs on each side of the array are purposed as Shift-Reg. As a result, only the central 20 × 20 PEs compute meaningful results. Without masking, it leads to an estimated 30.56% energy efficiency loss for the PE array. Through masking the boundary PEs, the power overhead can be reduced to only include their

TABLE I
POST-SYNTHESIS PPA COMPARISON FOR DIFFERENT PE DESIGNS

Specification	*2D-Mapping* PE	Systolic Conv. PE	Brute-Force Combination	Proposed PE w/o LRF and Mask
Area	$458\mu m^2$	$461\mu m^2$	$919\mu m^2$	$561\mu m^2$
Highest Frequency	1.4GHz	2.0GHz	1.4GHz	1.4GHz
Leakage	$0.51\mu W$	$0.51\mu W$	$1.02\mu W$	$0.6\mu W$
IPM Power	$26\mu W$	N/A	$26.5\mu W$	$26.4\mu W$
SCM Power	N/A	$16.5\mu W$	$17\mu W$	$17\mu W$

Type	Instruction	Description
RISC Control	BRCH/JP	Branch/jump
	SCALU	Scalar integer arithmetic
	SCLD/SCST	Scalar DMEM load/store to/from RISC core RF
	IMGSWP	Swap physical blocks in IAMEM
	SWMSK	Switch PE array to use the Msk-Reg as enable or not
	INTRQ	Interrupt external host
PE Array Arithmetic	IMGFILT	Filter image block in the PE array with specified weights from Filter RF
	ALUEW	Pixel-wise arithmetic with both inputs from PE's local RF
	ALUBC	Pixel-wise arithmetic with one input broadcast from RISC core RF
	ALUIMM	Pixel-wise arithmetic with one input broadcast from immediate
	ALUTR	Shift PE array data to ALU tree for reduction kernels
PE Array Data Movement	PESHFT	SREG data shifted by one-hop with specified direction
	MVREG	Move data among PE's local RF, SREG, PREG and Msk-Reg
	LD2D/ST2D	IAMEM load/store to/from SREG for 24×24 image blocks
	STVEC	Store masked SREG data to scalar DMEM as a vector
	LDMSK/ STMSK	Mask MEM load/store to/from Msk-Reg for 24×24 image blocks
Neural Network	NNRUN	Execute NN inference starting from specified 1st-layer in NN table

NN Layer	Description
CONV	Convolution layer
CONV_BN	Fused convolution and batch normalization
CONV_RS	Fused convolution and residual addition
CONV_RL	Fused convolution and ReLU
CONV_BNRL	Fused convolution, batch normalization and ReLU
CONV_RSRL	Fused convolution, residual addition and ReLU
CONV_UP	Up-sampling convolution
CONV_RLUP	Fused up-sampling convolution and ReLU
POOL_MAX	Max pooling
POOL_AVG	Average pooling
BNORM	Batch normalization
RES	Residual addition
RELU	ReLU
BNRL	Fused batch normalization and ReLU
RSRL	Fused residual addition and ReLU
FCL	Fully-connected layer

Fig. 7. ISA of NVPU showing major instructions.

leakage and SREG dynamic power. Post-automatic placement and routing (APR) simulation results of VGA Harris corner detection at 0.8 V and 100 MHz show that masking unused boundary PEs during image filtering and pixelwise operations help reduce the overall energy per frame of by 26%. In addition, for a VGA image with 60 features tracked from previous frames and a feature masking radius of 30 pixels, 43% of pixels can be masked for new feature detection on average (regions near old features are not considered). Gating their ALUs, LRFs, and Acc-Regs leads to a measured 17% power reduction for Harris corner detection at 0.52 V. Note that, the NVPU programs used for measurement results in Section V do not implement masking. Masking does not contribute to performance improvements; instead, it may result in lower utilization rate of the PE array.

C. Instruction Set Architecture

Fig. 7 shows the NVPU ISA, which combines aspects from various paradigms. It primarily follows the SIMD execution model. In terms of ISA flavor, it leans toward RISC but incorporates complex instruction set computer (CISC) features for dedicated high-efficiency acceleration of processing kernels. For instance, the IMGFILT instruction can be thought of as a microcode constructed from PESHFT and pixelwise ALU (MAC) operations, controlled by a dedicated finite state machine (FSM) within the RISC core decoder. In addition, higher level functions, such as CNN layers, are controlled by dedicated FSMs for optimized efficiency, as seen with

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

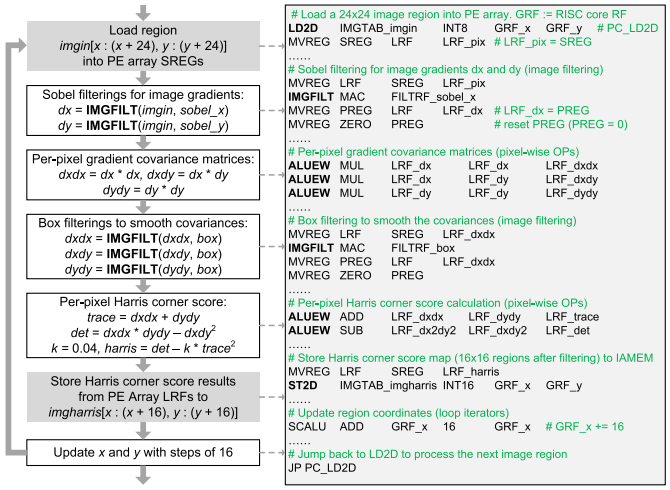


Fig. 8. Mapping Harris corner feature detection to NVPU ISA.

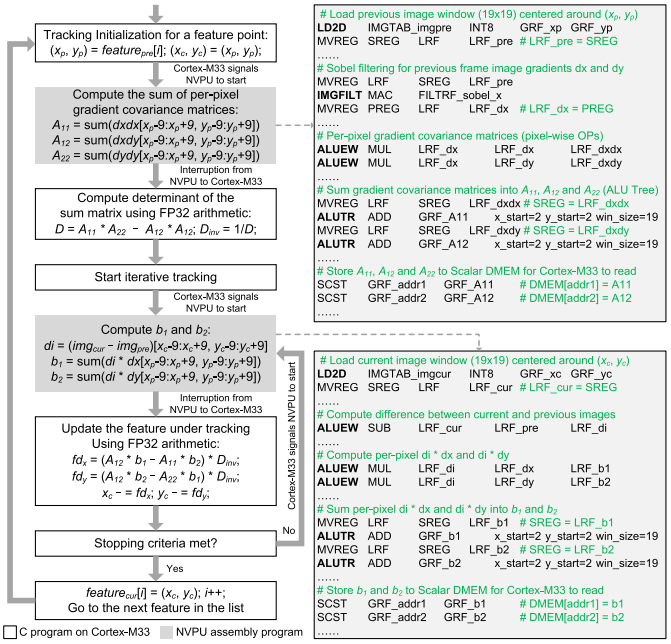


Fig. 9. Mapping sparse LK flow to Cortex-M33 and NVPU ISA.

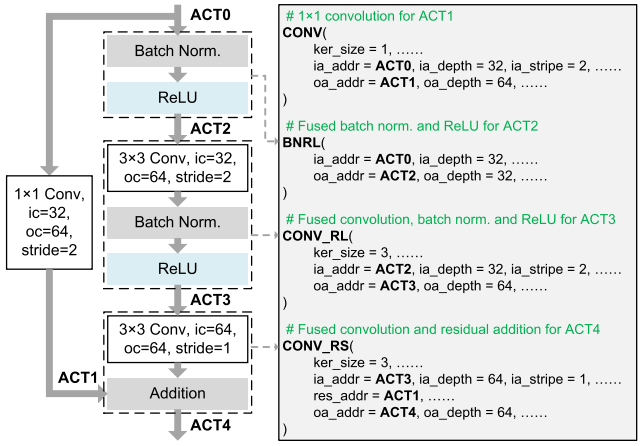


Fig. 10. Mapping full pre-activation residual block to NVPU ISA.

the NNRUN instruction and supported NN layers. The NN ISA exploits layer fusion between convolution and NN SIMD units, with layers, such as CONV_RS.[1] The NN ISA also supports high flexibility in configuring layer parameters, such as feature map and filter sizes, stride, decimal point positions, and memory addresses. NN layers are stored in the NN table and executed sequentially, starting from the specified layer in NNRUN and stops whenever the layer is marked as the last. This flexibility allows slicing large layers into smaller chunks for fused execution, reducing memory usage.

Fig. 8 shows an example of how the NVPU ISA is used to program Harris corner score calculation, the major step in Harris corner feature detection. In the example, images are processed at the macroblock (16 × 16 pixels) granularity, and intermediate result reusing enabled by the LRFs is exploited to minimize IAMEM load and store overhead. For each loop iteration, an image block is first loaded using LD2D; then, the pixels are moved from SREGs to LRFs using MVREG. Subsequently, Sobel filtering is applied using IMGFILT, with gradient results stored in LRFs. Per-pixel gradient covariance values are then calculated using pixelwise operations (ALUEW). Next, box filtering is applied to the covariance using IMGFILT again. After that, Harris corner scores are calculated using a series of pixelwise operations (ALUEW and ALUBC), with the parameter k broadcast from RISC core RF. Finally, Harris corner scores are stored to IAMEM, and the iteration advances onto the next macroblock. To further implement masking, mask bits for an image block can be loaded from the image mask memory into the PEs using LDMSK. Then, SWMSK is used to activate mask-based gating, before Sobel filtering starts. Custom masks for a block can be flexibly generated through PE arithmetic instructions and then stored to the image mask memory using STMSK.

Fig. 9 illustrates the mapping of sparse LK flow (with only one pyramid level in this example) onto RoboVisio, utilizing both Cortex-M33 and NVPU. The program flow is controlled by Cortex-M33. For each feature, Cortex-M33 first signals NVPU to compute A_{11}, A_{12}, and A_{22}, forming a matrix by summing the previous frame's per-pixel gradient covariance matrices within the local window (19 × 19 in this example) around the previous frame feature position (x_p and y_p). With the results, Cortex-M33 further computes the determinant D of that matrix and its reciprocal. Then, iterative tracking of the feature starts. In each iteration, Cortex-M33 signals NVPU to compute b_1 and b_2 based on the gradients of previous frame and the difference between the previous and current frame. Following that, Cortex-M33 computes the displacements fd_x and fd_y and updates the current frame feature position (x_c and y_c). At the end of each iteration, Cortex-M33 checks if stopping criteria have been met and decide whether to continue tracking or proceed to the next feature.

Fig. 10 shows an example of how a variation of the full pre-activation residual block [39] in ResNet [40] is mapped to NVPU's NN layers, which is the main building block of

[1]Note that, the prevalent practice today for applying post-convolution BN is to absorb BN parameters into the convolution layer via associativity of multiplication, instead of CONV_BN shown in Fig. 7.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

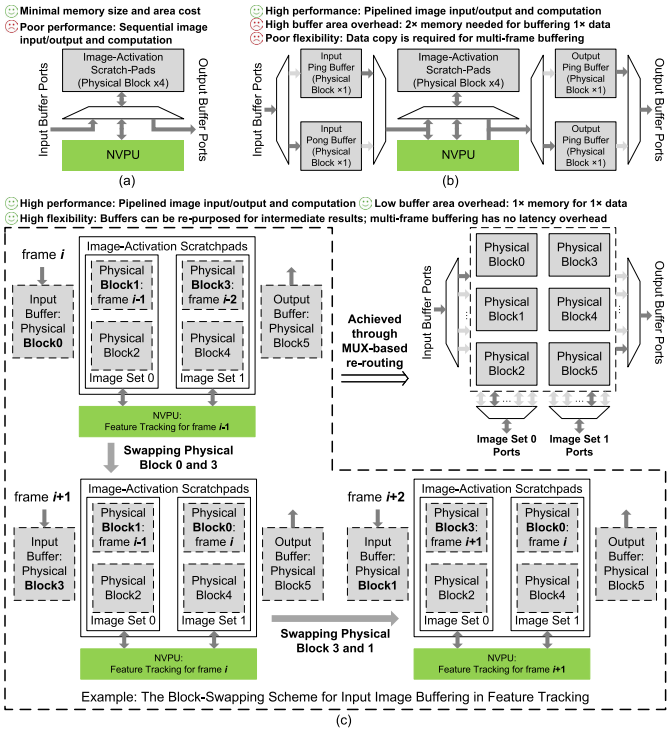


Fig. 11. Different buffering schemes for image input/output. (a) No buffering for image input/output. (b) Ping-pong buffering for image input/output. (c) Proposed block-swapping-based buffering for image input/output.

DroNet [15] for collision avoidance. Layer fusion between convolution and NN SIMD units is exploited in this case. Without fusion, the block requires seven layers to be computed with the storage of eight intermediate feature maps in IAMEM. Through using fusion layers (BNRL, CONV_RL, and CONV_RS), only four layers need to be computed with the storage of five intermediate feature maps. For a $24 \times 24 \times 32$ input feature map (ACT0), the memory footprint required for the residual block can be reduced by 30% with layer fusion, from 90 to only 63 kB.

IV. On-Chip Memory System Optimizations

A. Efficient Image Buffering Through Block Swapping

The design of input/output buffering is critical to the performance of vision processing systems. In Fig. 11, various buffering schemes for image input/output are compared. In the case of no buffering [Fig. 11(a)], minimal area cost can be achieved, but this leads to suboptimal performance due to the sequential execution of image input/output and computation. As illustrated in Fig. 11(b), ping-pong buffering addresses the performance issue by enabling parallel or pipelined image input/output and computation, a popular choice for conventional vision processors. However, this approach has inherent limitations. First, ping-pong buffering demands twice the amount of buffer memory to store a single piece of data, constituting a substantial area overhead. Second, buffer address space is typically isolated from that of the scratchpads, rendering buffers incapable of being repurposed for intermediate results. Furthermore, in vision tasks, such as feature tracking, there is a requirement to buffer and access multiple frames for computation. With ping-pong buffering, frames already buffered must be copied elsewhere to enable their reuse in subsequent computations and to free up buffers for incoming frames. This introduces significant latency overhead.

To overcome these limitations, we propose a unified IAMEM architecture with block-swapping-based buffering, where separation of buffer and scratchpad address spaces is only logical but not physical. Depicted in Fig. 11(c), it consists of physical memory blocks that can be swapped through MUX-based re-routing to any of the logical blocks in a single cycle. Specifically, for the RoboVisio test chip, six blocks are implemented. Logical block 0 is dedicated to input buffering, and logical block 5 handles output buffering, while the other four form two image sets that can be accessed simultaneously. Compared with the case of no buffering, this scheme enables pipelined image input/output and computation with a cost of only one memory block for either input or output. This represents a cost reduction by half compared with ping-pong buffering. This flexible scheme further avoids data copying in multi-frame buffering and allows buffers to be re-purposed for intermediate results. Fig. 11(c) illustrates the block swapping for feature tracking, where two frames (previous and current) need to be retained in the scratchpads. As NVPU processes frame $i-1$ and $i-2$, frame i is streamed into the input buffer. Once both the processing and streaming are completed, frame $i-2$ is no longer needed, and a swapping of physical blocks 0 and 3 is issued. Frame i then stays in the scratchpad and is ready for processing, while frame $i+1$ begins to be streamed in. After feature tracking for frame i is completed and frame $i+1$ is fully streamed, frame $i-1$ is no longer needed. A swapping between physical blocks 3 and 1 is issued, and frame $i+1$ is moved to the scratchpad, while frame $i+2$ starts to be streamed in.

B. MRAM Read Activity Minimization for CNN Inference

On-chip MRAM has emerged as a highly promising solution for retentive data storage in resource-constrained Internet of Things (IoT) devices [41], [42], [43], [44], mainly attributed to its ability in offering significantly higher area density than SRAM ($\sim 2\times$), while also delivering decent read speed (maximally $\sim$8-Gb/s bandwidth for the macro used by RoboVisio). In 22-nm CMOS, MRAM also provides $>35\%$ lower leakage power per bit than SRAM. However, MRAM circuits require a specific I/O voltage domain, ranging from 1.6 to 3.6 V. This characteristic gives rise to a notable dynamic power overhead, making MRAM read expensive in terms of energy. In fact, the energy per bit for MRAM reads can exceed that of SRAM by $>10\times$. Unfortunately, in SCM, weights are not stationary, and new weights must be streamed into the array every cycle to maintain full utilization. Thus, directly supplying weights from MRAM to NVPU leads to 100% MRAM read activity and high MRAM read power.

As depicted in Fig. 5, RoboVisio employs SRAM-based ping-pong weight buffers between MRAM and NVPU, where weights can be potentially reused, and MRAM might be read only intermittently. However, adding weight buffers alone does not help if the CNN loop order is suboptimal. When the loop

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

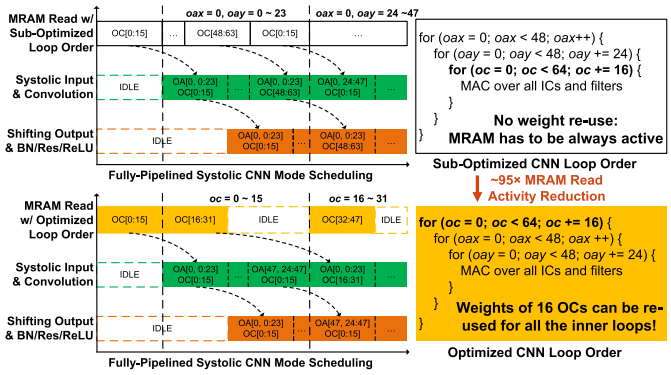
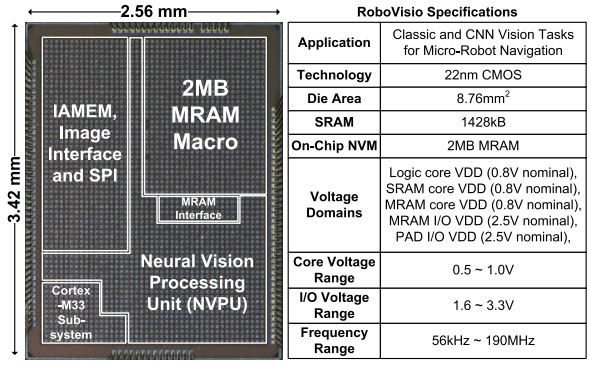


Fig. 12. MRAM read activity minimization with optimized CNN loop order.



RoboVisio Specifications	
Application	Classic and CNN Vision Tasks for Micro-Robot Navigation
Technology	22nm CMOS
Die Area	8.76mm^2
SRAM	1428kB
On-Chip NVM	2MB MRAM
Voltage Domains	Logic core VDD (0.8V nominal), SRAM core VDD (0.8V nominal), MRAM core VDD (0.8V nominal), MRAM I/O VDD (2.5V nominal), PAD I/O VDD (2.5V nominal),
Core Voltage Range	0.5 ~ 1.0V
I/O Voltage Range	1.6 ~ 3.3V
Frequency Range	56kHz ~ 190MHz

Fig. 13. Die photograph and specifications of RoboVisio.

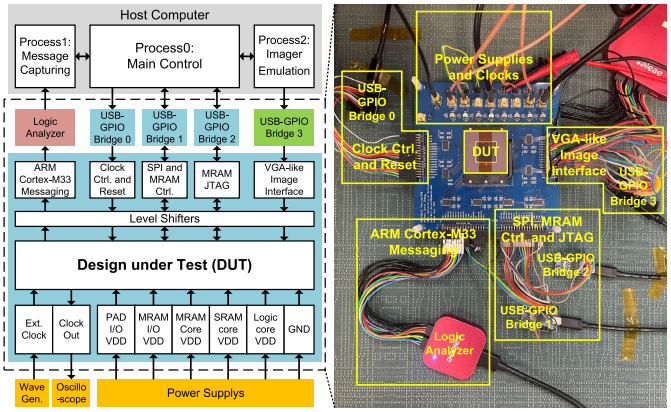


Fig. 14. Measurement setup for RoboVisio.

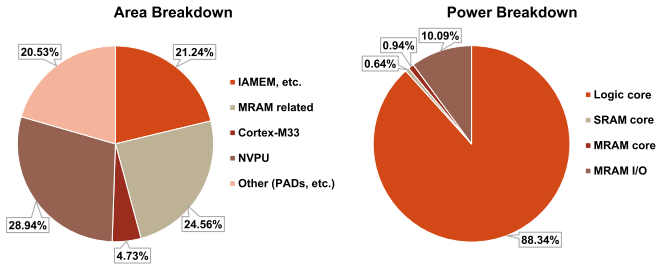


Fig. 15. Area and power breakdown of RoboVisio.

for output channels is placed as the innermost, as illustrated in Fig. 12 (top), the computation must iterate through all output channels before advancing to the next feature map region. Consequently, weights in the buffers cannot be reused, and MRAM is still read with 100% activity. MRAM bandwidth also becomes a limiting factor for computation throughput. As Fig. 12 (bottom) shows, RoboVisio addresses the challenge with an optimized loop order, where the loop for output channels is placed as the outermost. This enables the reuse of weights across 16 output channels for a complete output feature map, allowing MRAM to be read only once before the inner loops start. Following this initial MRAM read, weights are streamed from the buffer, and MRAM can remain inactive. Theoretically, this loop order minimizes MRAM read activity based on the SCM data flow. For the specific layer sizes in Fig. 12 ($96 \times 96 \times 16$ input to $48 \times 48 \times 64$ output with 3×3 filters), the optimized loop order leads to almost $95\times$ reduction in MRAM read activity, which has the potential to significantly decrease MRAM dynamic power.

V. Measurement Results

RoboVisio is implemented and fabricated in a 22-nm CMOS technology supporting embedded MRAM devices. Fig. 13 shows the die photograph and specifications of RoboVisio. It has a silicon footprint of 3.43×2.56 mm with an area of 8.76 mm^2. It has 2-MB MRAM and 1428-kB SRAM in total. Clock is supplied through an on-chip clock generator using a configurable ring oscillator and a configurable clock divider. The chip has five voltage domains: the logic core VDD for logic gates and SRAM peripherals (0.8-V nominal), the SRAM core VDD for bit cells (0.8-V nominal), the MRAM core VDD (0.8-V nominal), the MRAM I/O VDD (2.5-V nominal), and the pad I/O VDD (2.5-V nominal).

Fig. 14 shows the measurement setup for RoboVisio. The die uses the ceramic pin grid array (PGA) package. Sub-Miniature version A (SMA) cables and connectors are used for supplying voltages. SMAs are also used for supplying external debugging clock and for observing the on-chip clock through an oscilloscope. Level shifter chips are used to convert between standard I/O voltages (3.0/3.3 V) and RoboVisio's pad I/O voltage. The setup employs FTDI bridge boards that convert universal serial bus (USB) to general-purpose input/output (GPIO), enabling the host computer to directly and conveniently control RoboVisio through simple application programming interfaces (APIs) in Python. An FT2232H USB-GPIO bridge is used to emulate the imager that sends pixels to the RoboVisio's image interface using the VGA-like protocol. FT232H USB-GPIO bridges are used for the main process to control clock and reset I/Os, MRAM I/Os, and SPI I/Os. A logic analyzer is further automated to capture printed messages from the ARM Cortex-M33 core through a custom clocked interface (one character per cycle), making the testing of RoboVisio more interactive.

Fig. 15 shows the area and power breakdown of RoboVisio. Notably, we observe that the 2-MB MRAM occupies a similar area to that of the 936-kB IAMEM, highlighting MRAM's impressive area density, $\sim 2\times$ that of SRAM. The power breakdown is measured from a realistic scenario where RoboVisio attains high CNN energy efficiency during iterative execution of a convolution layer at 0.5-V core voltages, 1.6-V MRAM

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

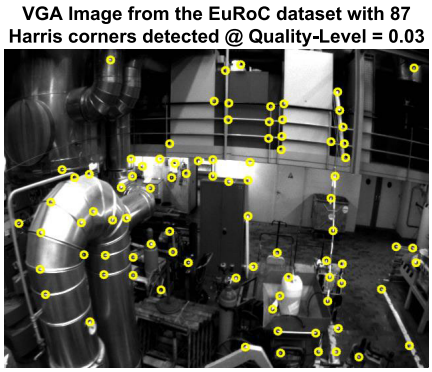


Fig. 16. Demonstration of Harris corner feature detection using RoboVisio.

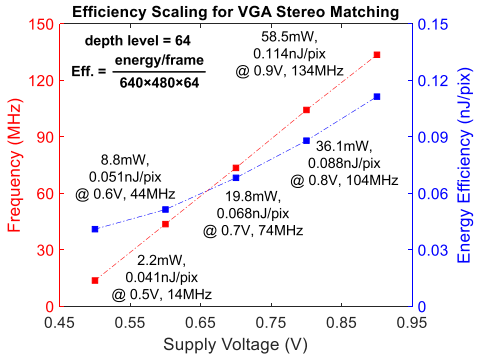


Fig. 17. Voltage, frequency, and efficiency scaling for VGA stereo matching.

I/O voltage, and 16 MHz. By employing the MRAM read activity minimization technique, the MRAM I/O domain only consumes ∼10% power. Otherwise, as will be discussed in Section V-B, MRAM can consume comparable or even higher power than logic and significantly degrade energy efficiency. In contrast, the MRAM core and SRAM core domains each consume only less than 1% of the total power.

A. Measurement Results for Classic Vision Tasks

Fig. 16 demonstrates Harris corner feature detection using NVPU's IPM. We apply feature detection to a VGA machine hall scene image from the EuRoC dataset [45], which is specialized for testing algorithms used in autonomous navigation, such as SLAM. In this experiment, we set the quality level to be 0.03, which signifies that only pixels with Harris corner scores $>0.03\times$ the maximum score are identified as features. As a result, 87 features are detected and highlighted in yellow circles. These features predominantly correspond to visually sharp corners within the scene and are well suited for feature tracking required by tasks, such as SLAM.

Fig. 17 shows the voltage, frequency, and energy efficiency scaling of RoboVisio when executing stereo local matching on VGA images using IPM. As IPM does not require MRAM, the measured power and energy efficiency for classic vision tasks do not include MRAM power. To calculate the energy efficiency of stereo local matching, the energy for processing a frame is first divided by the number of pixels in the frame. The result is then divided by the depth level (64 for the experiment). IPM frequency scales almost linearly with core voltages, and energy efficiency gets worse, as core voltages increases. In IPM, RoboVisio can operate at a lowest 0.5-V core voltage with a maximum frequency of 14 MHz and a highest 1.0-V core voltage with a maximum frequency of 180 MHz, where a peak IPM performance of 207 GOPS (16-bit OP) is achieved. For the primary classic vision tasks, RoboVisio achieves 0.22 nJ/pixels and 50 frames/s for VGA Harris corner feature detection at 0.52 V and 20 MHz, 0.22 nJ/pixels and 23 frames/s for VGA sparse LK flow (with $\leq$100 features per frame) at 0.52 V and 20 MHz, and 0.055 nJ/pixels and 10 frames/s for VGA stereo local matching at 0.62 V and 50 MHz.

TABLE II
SUMMARY OF DRONET PERFORMANCE AND EFFICIENCY

Application	Learning-based Collision Avoidance
NN Architecture	ResNet-8 variant with full pre-activation residual blocks
Input Size	200×200
Output (Regression results)	P_{coll}: collision probability θ_{steer}: steering angle to avoid collision
Accuracy (from [15])	Collision classification accuracy: 95.4% Steering angle regression RMSE: 6.25°
Parameter Size	337KB pre-trained model (8-bit quantized) mapped fully on MRAM
Best Efficiency on RoboVisio	53fps, 2.6mW and 49.06μJ/frame, @ 0.52V Core, 1.6V MRAM I/O and 18MHz
Highest Performance on RoboVisio	554fps, 78.38mW and 141.5μJ/frame, @ 1.0V Core, 1.6V MRAM I/O and 187MHz

B. Measurement Results for CNN

First, end-to-end CNN inference on RoboVisio is evaluated using DroNet [15], a ResNet-8 variant for learning-based collision avoidance where the full pre-activation residual blocks [39] discussed in Fig. 10 are used. Table II provides a summary of DroNet performance and efficiency when executed on RoboVisio. The input image size for DroNet is 200 × 200. DroNet produces regression results: the collision probability P_{coll} and the steering angle θ_{steer}, which guides robot motion to avoid obstacles. The floating-point DroNet model [15] achieves 95.4% accuracy for collision classification and 6.25° root-mean-square error (RMSE) for steering angle regression. With 8-bit quantization, 16-bit MAC, and full support for tuning per-tensor decimal point positions, RoboVisio is expected to achieve less than 0.2% loss for those metrics, according to post-training quantization accuracy of ResNet from [46]. Parameter size of the 8-bit DroNet model is 337 kB, which can be fully mapped onto the on-chip MRAM. For the best DroNet energy efficiency, RoboVisio achieves 49.06 μJ/frame at 0.52-V core voltages, 1.6-V MRAM I/O voltage, and 18 MHz, with 53 frames/s and 2.6 mW. For the highest performance, RoboVisio reaches 554 frames/s at 1.0-V core voltages, 1.6-V MRAM I/O voltage, and 187 MHz (dual-cycle MRAM read), with 78.38 mW and 141.5 μJ/frame. Table III further details the sparsity, latency, and energy efficiency breakdown for DroNet layers at 0.52 V and 18 MHz. RoboVisio attains the highest per-layer efficiency of 3.01 TOPS/W (16-bit OP) for a CONV_RS layer. Overall, RoboVisio achieves 1.69 TOPS/W (16-bit OP) for DroNet inference at 0.52 V and 18 MHz.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

TABLE III
SPARSITY, LATENCY, AND EFFICIENCY BREAKDOWN FOR DRONET LAYERS

	Layer	Input Size	Ouput Size	k, s	[1]Input Activation Sparsity	Weight Sparsity	Latency	Eff. (16-bit OP)
1	CONV	(200, 200, 1)	(100, 100, 32)	5×5, 2	0.03%	4.25%	3.17ms	1.39TOPS/W
2	POOL_MAX	(100, 100, 32)	(49, 49, 32)	3×3, 2	7.47%	N/A	2.90ms	0.12TOPS/W
3	CONV	(49, 49, 32)	(25, 25, 32)	1×1, 2	7.31%	6.54%	0.28ms	1.15TOPS/W
4	BNRL	(49, 49, 32)	(49, 49, 32)	N/A	7.31%	N/A	0.27ms	0.25TOPS/W
5	CONV_BNRL	(49, 49, 32)	(25, 25, 32)	3×3, 2	57.97%	73.78%	1.66ms	2.79TOPS/W
6	CONV_RS	(25, 25, 32)	(25, 25, 32)	3×3, 1	85.56%	62.0%	1.66ms	2.70TOPS/W
7	CONV	(25, 25, 32)	(13, 13, 64)	1×1, 2	6.01%	6.88%	0.15ms	1.11TOPS/W
8	BNRL	(25, 25, 32)	(25, 25, 32)	N/A	6.01%	N/A	0.09ms	0.20TOPS/W
9	CONV_BNRL	(25, 25, 32)	(13, 13, 64)	3×3, 2	78.24%	85.81%	0.87ms	2.89TOPS/W
10	CONV_RS	(13, 13, 64)	(13, 13, 64)	3×3, 1	95.46%	78.65%	1.73ms	3.01TOPS/W
11	CONV	(13, 13, 64)	(7, 7, 128)	1×1, 2	6.33%	3.04%	0.22ms	0.87TOPS/W
12	BNRL	(13, 13, 64)	(13, 13, 64)	N/A	6.33%	N/A	0.05ms	0.24TOPS/W
13	CONV_BNRL	(13, 13, 64)	(7, 7, 128)	3×3, 2	89.12%	95.18%	1.85ms	1.74TOPS/W
14	CONV_RSRL	(7, 7, 128)	(7, 7, 128)	3×3, 1	96.46%	98.43%	3.69ms	1.71TOPS/W
15	FCL	6272	1	N/A	98.95%	59.47%	0.10ms	0.04TOPS/W
16	FCL	6272	1	N/A	98.95%	60.25%	0.10ms	0.04TOPS/W
Overall					N/A	N/A	18.8ms	1.69TOPS/W

[1]The input activation sparsity shown here is from the specific input in Fig. 18.

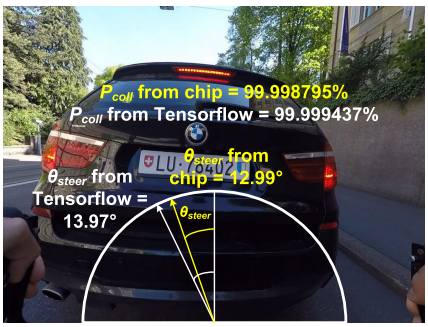


Fig. 18. Demonstration of DroNet-based collision avoidance using RoboVisio.

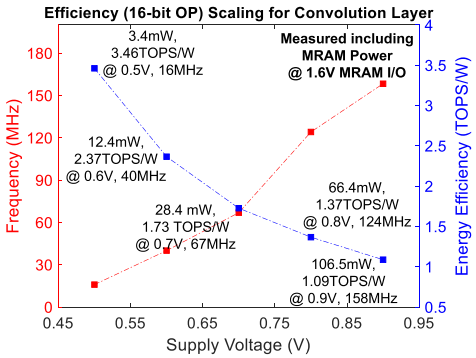


Fig. 19. Scaling for a convolution layer (80% wt. sparsity and 50% act. sparsity).

Fig. 18 presents a demonstration of DroNet using RoboVisio, where the robot is approaching a car, leading to a potential collision. The result from TensorFlow (floating-point software) and the result from RoboVisio both give >99.99% collision probability, with <0.001% numerical difference. Furthermore, they both suggest steering angles that direct the robot leftward to avoid the collision, with only <1° numerical difference. These minute numerical differences have negligible impacts on the robot's motion.

We further assess RoboVisio's peak SCM energy efficiency using a synthetic convolution layer. To maximally utilize the PE array, we configure the layer with 48 × 48 × 32 input, 24 × 24 × 32 output, and 3 × 3 filters. In addition, the layer is set with 80% weight sparsity, 50% input activation sparsity, and random input values. According to Table III, the chosen sparsity levels are significantly lower than those observed in DroNet's sparsest convolution layer (layer 14). Therefore, the synthetic layer serves as a conservative estimation of the maximally achievable energy efficiency under realistic scenarios. Fig. 19 shows the voltage, frequency, and energy efficiency scaling for this experiment. The measured power and energy efficiency for CNN include MRAM powers, with 1.6-V MRAM I/O voltage. In SCM, RoboVisio can operate at a lowest 0.5-V core voltage, achieving a maximum frequency of 16 MHz and an energy efficiency of 3.5 TOPS/W (16-bit OP). At the highest 1.0-V core voltage, it reaches a maximum frequency of 190 MHz, delivering a peak SCM performance of 146 GOPS (16-bit OP). Due to the constant MRAM I/O voltage, the scaling of SCM frequency demonstrates less linearity than that of IPM. SCM energy efficiency gets worse, as core voltages increases.

Fig. 20 shows measurement results for RoboVisio's CNN weight memory system, including MRAM and ping-pong weight buffers. The figure illustrates the effect of the MRAM read activity minimization technique introduced in Section IV-B. MRAM read powers are measured at 1.6-V MRAM I/O voltage with varying MRAM core voltages (0.7, 0.8, and 0.9 V). Corresponding powers for SRAM-based ping-pong weight buffers are obtained from the SRAM compiler. To characterize the power of the weight memory system without optimizations, MRAM read power at 100% activity is measured through continuously reading the MRAM using Cortex-M33. Power for the weight memory system employing the activity minimization technique (the case for RoboVisio) is measured during iterative execution of a convolution layer (48 × 48 × 32 input, 24 × 24 × 32 output, and 3 × 3 filters) with fully dense and random weights. Across different core voltages, frequencies, and read cycles, significant power reductions are observed. Notably, at the nominal 0.8-V core voltage and 110 MHz with dual-cycle MRAM read, the proposed technique achieves an 75% reduction in overall weight

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
CHIP SUMMARY AND COMPARISON WITH STATE-OF-THE-ART

	VLSI '17 [20]	JSSC '19 [16]	ISSCC '19 [27]	ISSCC '21 [41]	JSSC '22 [47]	JSSC '23 [44]	[1]ISSCC '23 [48]	This work
Techonology	28nm CMOS	65nm CMOS	22nm FDSOI	22nm FDSOI	40nm CMOS	22nm FDSOI	22nm FDSOI	22nm CMOS
Application	Multi-View Depth	Visual-Inertial Odometry	Image Processing	IoT NN Inference	Edge NN Inference and Weight Tuning	IoT ML Inference	IoT Artificial Intelligence	Classic and CNN Vision Tasks for Micro-Robot Navigation
Processing Architecture	Dedicated Accelerator	Dedicated Accelerator	Streaming-based CGRA	CMP and NN Accelerator	Systolic Convolution PE Array	2D SIMD PE Array	CMP and NN Accelerator	Hybrid Systolic 2D-Mapping PE Array
Programmability	Not Programmable	Not Programmable	Only for Image Processing	General Purpose and NN	Only for NN	General Purpose and ML	General Purpose and NN	General Purpose, Image Processing and NN
Die Area	5.96mm^2	20mm^2	4.9mm^2	12mm^2	29.2mm^2	6.25mm^2	18.7mm^2	8.76mm^2
SRAM	582.5kB	854kB	690kB	1728kB	0.5MB	708kB	1152kB	1428kB
On-Chip NVM	N/A	N/A	N/A	4MB MRAM	2MB RRAM	0.5MB MRAM	N/A	2MB MRAM
Voltage	0.9V	1.0V	0.8V	0.5 ∼ 0.8V	1.1V	0.4 ∼ 0.9V	0.5 ∼ 0.8V	0.5 ∼ 1.0V
Max. Frequency	300MHz	62.5MHz	220MHz	450MHz	200MHz	150MHz	420MHz	190MHz
Power	380mW	24mW	10.7 ∼ 101.4mW	1.7μW ∼ 49.4mW	126mW	1.7μW ∼ 20mW	12.8mW ∼ 123mW	468μW ∼ 158mW
Peak Performance	Not Reported	[2]59.1GOPS	[3]145GOPS	[4]32.2GOPS (NN Accelerator)	[4]920GOPS	[4]17.6GOPS	[4]122GOPS, [5]637GOPS (NN Accelerator)	[3]207GOPS
NN Inference Efficiency	N/A	N/A	Not Reported	[4,6]1.3TOPS/W (NN Accelerator)	[4,6]2.2TOPS/W	[4,6]2.5TOPS/W	[4,6]1.7TOPS/W, [5,6]12.4TOPS/W (NN Accelerator)	[3,7]3.5TOPS/W, [3,8]3.01TOPS/W

[1]8-bit results are inferred from Fig. 22.1.3 and Fig. 22.1.5 of [48]. [2]Arithmetic precision not specified. [3]16-bit MAC. [4]8-bit MAC. [5]2-bit MAC. [6]Sparsity not specified. [7]80% weight sparsity and 50% input activation sparsity. [8]Highest per-layer energy efficiency for DroNet inference on RoboVisio.

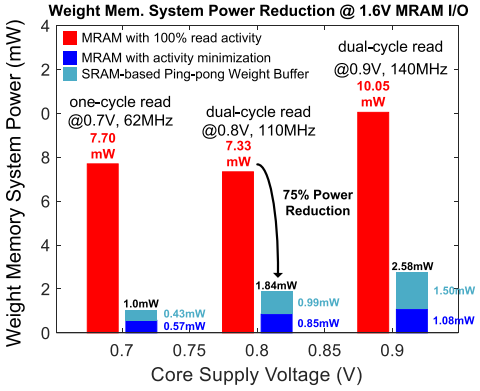


Fig. 20. CNN weight memory system power reduction.

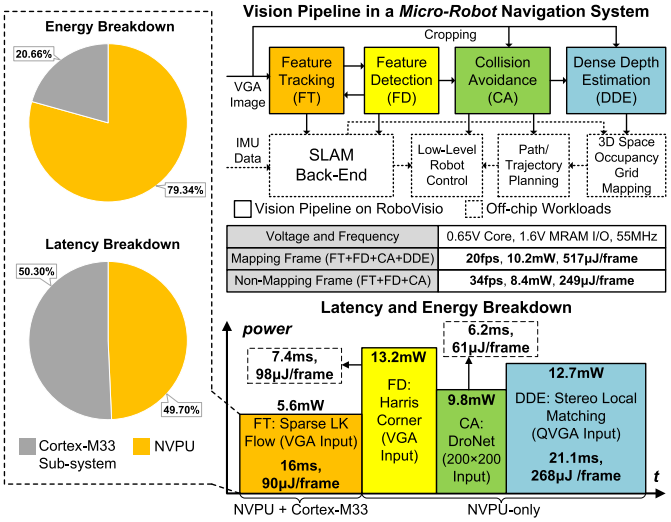


Fig. 21. Evaluation of the complete vision pipeline for micro-robot navigation.

memory system power. Intriguingly, after activity minimization, MRAM read power is reduced to a comparable level as that of the SRAM-based weight buffers.

C. System Evaluation for the Navigation Vision Pipeline

As Fig. 21 shows, we further evaluate a complete vision pipeline in a micro-robot autonomous navigation system on RoboVisio. The non-vision workloads are expected to be executed off-chip. The vision pipeline employs Harris corner for feature detection, sparse LK flow [11] for feature tracking, stereo local matching for depth estimation, and DroNet for collision avoidance. Feature detection and tracking process VGA input images, which are cropped to quarter VGA (QVGA) images for stereo local matching and 200 × 200 images for collision avoidance. Cropping can be supported by IAMEM load and store without energy and latency overhead. RoboVisio is setup at 0.65-V core voltages, 1.6-V MRAM I/O voltage, and 55 MHz for this evaluation. As 3-D mapping does not necessarily require dense depth for every frame, two different scenarios are measured: the mapping frames that compute depth, and the non-mapping frames that do not compute depth. For mapping frames, 20 frames/s can be achieved with 10.2-mW power and 517-μJ/frame energy efficiency. For non-mapping frames, 34 frames/s can be achieved with 8.4 mW and 249-μJ/frame energy efficiency. In a more realistic scenario where depth is only computed at 5 frames/s (with the other tasks at 20 frames/s), RoboVisio power is estimated to be only 6.3 mW. This suggests that for vision tasks in micro-robot autonomy, the sub-10-mW power target can be successfully met using RoboVisio. Fig. 21 also provides the energy and latency breakdown for the four vision tasks, where stereo local matching emerges as the bottleneck in both performance and efficiency. Out of the four tasks, only sparse LK flow involves the use of Cortex-M33 sub-system. The sparse LK flow energy and latency breakdown for NVPU and Cortex-M33 sub-system are shown in Fig. 21 (left).

To further provide on-chip support for the entire micro-robot navigation system, a future design can extend RoboVisio

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

TABLE V
COMPARISON FOR CLASSIC VISION TASKS

	Feature Detection	Feature Tracking	Dense Depth Estimation
[1]ARM Cortex-M33	4.36nJ/pix, 0.18mW, 0.13fps VGA, Harris corner @ 0.52V, 20MHz	0.37nJ/pix, 0.18mW, 1.53fps VGA, sparse LK flow ($\leq$ 100 features) @ 0.52V, 20MHz	2.43nJ/pix, 0.62mW, 0.013fps VGA, stereo local matching (depth level = 64) @ 0.62V, 50MHz
VLSI '17 [20]	N/A	N/A	0.042nJ/pix, 32fps 2K, stereo local matching
JSSC '19 [16]	[2]0.33nJ/pix, 71fps Wide-VGA (752x480), Shi-Tomasi corner	[2]1.40nJ/pix, 71fps Wide-VGA (752x480), sparse LK flow	N/A
ISSCC '19 [27]	Not Reported	1.16nJ/pix, 30fps VGA, dense LK flow	Not Reported
This work	0.22nJ/pix, 3.4mW, 50fps VGA, Harris corner @ 0.52V, 20MHz	0.22nJ/pix, 1.6mW, 23fps VGA, sparse LK flow ($\leq$ 100 features) @ 0.52V, 20MHz	0.055nJ/pix, 10.9mW, 10fps VGA, stereo local matching (depth level = 64) @ 0.62V, 50MHz

[1]Post-APR simulation in 22nm. [2]Energies calculated from Fig. 11 and Table III of [16].

to integrate additional blocks into the SoC. Specifically, for tasks, such as SLAM backend, it would be necessary to incorporate a linear algebra processor capable of computing matrix de-compositions and general matrix operations (addition, multiplication, and so on), similar to the one proposed in [22]. In addition, dedicated accelerators for specialized functions, such as the ones for occupancy grid mapping [49] and path planning [50], could be integrated to optimize the overall efficiency and performance of the system.

D. Comparison With State-of-the-Art

Table IV provides a summary of RoboVisio and compares it against state-of-the-art designs in terms of general properties and edge ML performance. In addition, Table V extends the comparison to encompass classic vision tasks, with ARM Cortex-M33 serving as the baseline. RoboVisio stands out by offering the broadest range of programmability, accommodating general-purpose processing, programmable image processing, and NN inference. Despite its high flexibility, RoboVisio demonstrates either superior or comparable energy efficiency when compared with dedicated accelerators. For CNN inference, RoboVisio achieves an impressive efficiency of 3.5 TOPS/W (16-bit OP). Though RoboVisio uses 16-bit MACs that exhibit considerably higher power than 8-bit MACs, it outperforms state-of-the-art NVM-based ML SoCs (all using 8-bit MACs) [41], [44], [47] by achieving energy efficiency improvements ranging from 40% to 170%, showing the potential advantage of RoboVisio's architecture.

VI. CONCLUSION

This article introduces RoboVisio, a highly efficient and flexible domain-specific SoC tailored for vision tasks in fully autonomous micro-robot navigation. Addressing the four primary vision tasks sequentially executed in a navigation vision pipeline, we propose a novel NVPU featuring a hybrid PE. Classic vision tasks benefit from a *2-D-mapping* architecture, while CNNs are efficiently executed in an *output-channel-parallel* systolic manner. By combining both processing schemes into a single PE array, our architecture ensures future compatibility with CNN-heavy vision algorithms, while saving 40% of area and leakage power compared with two separate array implementations, without power overhead and throughput loss. RoboVisio further optimizes energy and area efficiency through several key features: 1) integration of 2-MB MRAM for non-volatile fully-on-chip weight storage; 2) a unified IAMEM with block-swapping-based image buffering that reduces buffer size by 50% and eliminates data copy for multi-frame buffering; and 3) a combination of weight buffering and CNN loop ordering that reduces measured weight memory system power by 75%. RoboVisio is fabricated in 22-nm CMOS. For classic vision tasks, RoboVisio achieves high energy efficiencies that rival dedicated accelerators. For CNN inference, it achieves a highest efficiency of 3.5 TOPS/W (16-bit OP), surpassing state-of-the-art NVM-based edge ML SoCs by a remarkable 40%–170%.

ACKNOWLEDGMENT

The authors thank Arm Ltd., Cambridge, U.K., for supporting this work and the TSMC University Shuttle Program for chip fabrication.

REFERENCES

[1] F. Arvin, J. Murray, C. Zhang, and S. Yue, "Colias: An autonomous micro robot for swarm robotic applications," *Int. J. Adv. Robotic Syst.*, vol. 11, no. 7, p. 113, Jul. 2014, doi: 10.5772/58730.

[2] B. Goldberg et al., "Power and control autonomy for high-speed locomotion with an insect-scale legged robot," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 987–993, Apr. 2018, doi: 10.1109/LRA.2018.2793355.

[3] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," *Nature*, vol. 570, no. 7762, pp. 491–495, Jun. 2019, doi: 10.1038/s41586-019-1322-0.

[4] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle," *J. Field Robot.*, vol. 33, no. 4, pp. 431–450, Jun. 2016, doi: 10.1002/rob.21581.

[5] Y. Lin et al., "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, Jan. 2018, doi: 10.1002/rob.21732.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[7] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

[8] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, Manchester, U.K., vol. 15, 1988, pp. 147–151. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=88cdfbeb78058e0eb2613e79d1818c567f0920e2A

[9] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Seattle, WA, USA, Jun. 1994, pp. 593–600, doi: 10.1109/CVPR.1994.323794.

[10] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, Vancouver, BC, Canada, vol. 2, 1981, pp. 674–679, doi: 10.5555/1623264.1623280.

[11] J.-Y. Bouguet, "Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm," *Intel Corp.*, vol. 5, nos. 1–10, p. 4, 2001. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf

[12] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, pp. 7–42, Apr. 2002, doi: 10.1023/A:1014573219977.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

[13] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "FastDepth: Fast monocular depth estimation on embedded systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, Montreal, QC, Canada, May 2019, pp. 6101–6108, doi: 10.1109/icra.2019.8794182.

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013, doi: 10.1007/s10514-012-9321-0.

[15] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018, doi: 10.1109/LRA.2018.2795643.

[16] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mW fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE J. Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, Apr. 2019, doi: 10.1109/JSSC.2018.2886342.

[17] D. Jeon, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, "A 470 mV 2.7 mW feature extraction-accelerator for micro-autonomous vehicle navigation in 28 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2013, pp. 166–167, doi: 10.1109/ISSCC.2013.6487684.

[18] Z. Li, J. Wang, D. Sylvester, D. Blaauw, and H.-S. Kim, "A1920×1080 25FPS, 2.4TOPS/W unified optical flow and depth 6D vision processor for energy-efficient, low power autonomous navigation," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 135–136, doi: 10.1109/VLSIC.2018.8502347.

[19] Z. Li et al., "3.7 A 1920×1080 30fps 2.3TOPS/W stereo-depth processor for robust autonomous navigation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2017, pp. 62–63, doi: 10.1109/ISSCC.2017.7870261.

[20] J. Narinx, T. Demirci, A. Akin, and Y. Leblebici, "A single-chip 2048×1080 resolution 32 fps 380 mW trinocular disparity estimation processor in 28 nm CMOS technology," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, 2017, pp. C228–C229, doi: 10.23919/VLSIC.2017.8008489.

[21] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A fully integrated energy-efficient visual-inertial odometry accelerator for autonomous navigation of nano drones," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2018, pp. 133–134, doi: 10.1109/VLSIC.2018.8502279.

[22] Z. Li et al., "An 879GOPS 243 mW 80fps VGA fully visual CNN-SLAM processor for wide-range autonomous exploration," in *Proc. IEEE Int. Solid- State Circuits Conf. - (ISSCC)*, San Francisco, CA, USA, Feb. 2019, pp. 134–136, doi: 10.1109/ISSCC.2019.8662397.

[23] X. Zhang et al., "A fully integrated battery-powered system-on-chip in 40-nm CMOS for closed-loop control of insect-scale pico-aerial vehicle," *IEEE J. Solid-State Circuits*, vol. 52, no. 9, pp. 2374–2387, Sep. 2017, doi: 10.1109/JSSC.2017.2705170.

[24] V. Honkote et al., "2.4 A distributed autonomous and collaborative multi-robot system featuring a low-power robot SoC in 22 nm CMOS for integrated battery-powered minibots," in *ISSCC*, San Francisco, CA, USA, Feb. 2019, pp. 48–50, doi: 10.1109/ISSCC.2019.8662463.

[25] D. Moloney, B. Barry, R. Richmond, F. Connor, C. Brick, and D. Donohoe, "Myriad 2: Eye of the computational vision storm," in *Proc. IEEE Hot Chips 26 Symp. (HCS)*, Cupertino, CA, USA, Aug. 2014, pp. 1–18, doi: 10.1109/HOTCHIPS.2014.7478823.

[26] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. Horowitz, "Convolution engine: Balancing efficiency and flexibility in specialized computing," *Commun. ACM*, vol. 58, no. 4, pp. 85–93, Mar. 2015, doi: 10.1145/2735841.

[27] S. Smets, T. Goedemé, A. Mittal, and M. Verhelst, "2.2 A 978GOPS/W flexible streaming processor for real-time image processing applications in 22 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 44–46, doi: 10.1109/ISSCC.2019.8662346.

[28] Z. Wang et al., "An all-weights-on-chip dnn accelerator in 22nm ull featuring 24×1 mb erram" in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, 2020, pp. 1–2, doi: 10.1109/VLSICircuits18222.2020.9162811.

[29] Z. Li et al., "RRAM-DNN: An RRAM and model-compression empowered all-weights-on-chip DNN accelerator," *IEEE J. Solid-State Circuits*, vol. 56, no. 4, pp. 1105–1115, Apr. 2021, doi: 10.1109/JSSC.2020.3045369.

[30] H. An et al., "A 170μW image signal processor enabling hierarchical image recognition for intelligence at the edge," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2020, pp. 1–2, doi: 10.1109/VLSICircuits18222.2020.9162810.

[31] H. An et al., "An ultra-low-power image signal processor for hierarchical image recognition with deep neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 4, pp. 1071–1081, Apr. 2021, doi: 10.1109/JSSC.2020.3041858.

[32] H. An et al., "29.3 An 8.09TOPS/W neural engine leveraging bit-sparsified sign-magnitude multiplications and dual adder trees," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 422–424, doi: 10.1109/isscc42615.2023.10067269.

[33] M. Giordano et al., "CHIMERA: A 0.92 TOPS, 2.2 TOPS/W edge AI accelerator with 2 MByte on-chip foundry resistive RAM for efficient training and inference," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, Jun. 2021, pp. 1–2, doi: 10.23919/VLSICircuits52068.2021.9492347.

[34] J. Redgrave, A. Meixner, N. Goulding-Hotta, A. Vasilyev, and O. Shacham, "Pixel visual core: Google's fully programmable image vision and AI processor for mobile devices," in *Proc. IEEE Hot Chips 30 Symp.*, Cupertino, California, USA, Aug. 2018, pp. 1–18. [Online]. Available: https://old.hotchips.org/hc30/1conf/1.02_Google_HC30.Google.JasonRedgrave.V01.pdf

[35] Z. Du et al., "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Portland, OR, USA, Jun. 2015, pp. 92–104, doi: 10.1145/2749469.2750389.

[36] X. Wei et al., "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in *Proc. 54th Annu. Design Automat. Conf.*, Austin, TX, USA, 2017, pp. 1–6, doi: 10.1145/3061639.3062207.

[37] C. Sakr, S. Dai, R. Venkatesan, B. Zimmer, W. Dally, and B. Khailany, "Optimal clipping and magnitude-aware differentiation for improved quantization-aware training," in *Proc. 39th Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2022, pp. 19123–19138. [Online]. Available: https://proceedings.mlr.press/v162/sakr22a.html

[38] B. Keller et al., "A 17–95.6 TOPS/W deep learning inference accelerator with per-vector scaled 4-bit quantization for transformers in 5 nm," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Honolulu, HI, USA, Jun. 2022, pp. 16–17, doi: 10.1109/VLSITechnologyandCir46769.2022.9830277.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. 14th Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 630–645, doi: 10.1007/978-3-319-46493-0_38.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[41] D. Rossi et al., "4.4 A 1.3TOPS/W @ 32GOPS fully integrated 10-core SoC for IoT end-nodes with 1.7μW cognitive wake-up from MRAM-based state-retentive sleep mode," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2021, pp. 60–62, doi: 10.1109/ISSCC42613.2021.93659394.4.

[42] Q. Zhang et al., "A 22 nm 3.5 TOPS/W flexible micro-robotic vision SoC with 2MB eMRAM for fully-on-chip intelligence," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Honolulu, HI, USA, Jun. 2022, pp. 72–73, doi: 10.1109/VLSITechnologyandCir46769.2022.9830340.

[43] Z. Fan et al., "Audio and image cross-modal intelligence via a 10TOPS/W 22 nm SoC with back-propagation and dynamic power gating," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Honolulu, HI, USA, Jun. 2022, pp. 18–19, doi: 10.1109/VLSITechnologyandCir46769.2022.9830226.

[44] V. Jain, S. Giraldo, J. D. Roose, L. Mei, B. Boons, and M. Verhelst, "TinyVers: A tiny versatile System-on-Chip with state-retentive eMRAM for ML inference at the extreme edge," *IEEE J. Solid-State Circuits*, vol. 58, no. 8, pp. 2360–2371, Aug. 2023, doi: 10.1109/JSSC.2023.3236566.

[45] M. Burri et al., "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016, doi: 10.1177/0278364915620033.

[46] M. Nagel, M. Fournarakis, R. Ali Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," 2021, *arXiv:2106.08295*.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.

[47] K. Prabhu et al., "CHIMERA: A 0.92-TOPS, 2.2-TOPS/W edge AI accelerator with 2-MByte on-chip foundry resistive RAM for efficient training and inference," *IEEE J. Solid-State Circuits*, vol. 57, no. 4, pp. 1013–1026, Apr. 2022, doi: 10.1109/JSSC.2022.3140753.

[48] F. Conti et al., "22.1 A 12.4TOPS/W @ 136GOPS AI-IoT system-on-chip with 16 RISC-V, 2-to-8b precision-scalable DNN acceleration and 30%-boost adaptive body biasing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 21–23, doi: 10.1109/ISSCC42615.2023.10067643.

[49] T. Jia et al., "OMU: A probabilistic 3D occupancy mapping accelerator for real-time OctoMap at the edge," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Antwerp, Belgium, Mar. 2022, pp. 909–914, doi: 10.23919/date54114.2022.9774508.

[50] C. Chung and C.-H. Yang, "A 1.5-μJ/task path-planning processor for 2-D/3-D autonomous navigation of microrobots," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 112–122, Jan. 2021, doi: 10.1109/JSSC.2020.3037138.



Qirui Zhang (Member, IEEE) received the B.S. degree (Hons.) from the School of Microelectronics, Shanghai Jiao Tong University, Shanghai, China, in 2018. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Michigan, Ann Arbor, MI, USA.

His research interests are in efficient algorithm-hardware co-designs, very large-scale integration (VLSI) architectures, and integrated circuits for emerging applications, including artificial intelligence and quantum computing.

Mr. Zhang was a recipient of the Best Paper Award at the 2022 tinyML Research Symposium and the 2023 IEEE International Conference on Application-Specific Systems, Architectures and Processors. He was a Finalist of the 2023 Qualcomm Innovation Fellowship (North America).



Zichen Fan (Graduate Student Member, IEEE) received the B.S. degree from Tsinghua University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the Michigan Integrated Circuit Laboratory, University of Michigan, Ann Arbor, MI, USA.

His current research interests include machine learning accelerator design, algorithm-hardware co-design, and energy-efficient digital system design.



Hyochan An received the B.S. degree in electrical and computer engineering from Sungkyunkwan University, Seoul, South Korea, in 2014, and the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 2022.

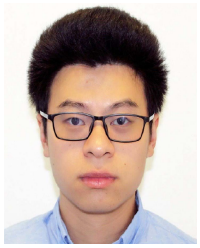
Between 2014 and 2017, he was an Engineer with Samsung Electronics, Hwaseong, South Korea. His current research interests include energy-efficient accelerator design and system.

Dr. An was a recipient of the Doctoral Fellowship from the Kwanjeong Educational Foundation in South Korea.



Zhehong Wang received the B.E. in electronics and information engineering from Zhejiang University, Hangzhou, China, in 2016, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2019 and 2022, respectively.

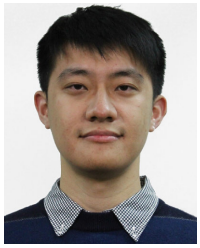
He is currently an ASIC Engineer at the Reality Labs, Meta Platforms Inc., Menlo Park, CA, USA, working on high-efficiency machine learning accelerators for AR/VR edge devices.



Ziyun Li (Member, IEEE) received the B.S. degree in electrical and computer engineering and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014 and 2019, respectively.

He is currently with Meta, Redmond, WA, USA. His research interests include SW and HW co-design of high-performance, energy-efficient AI systems and intelligent cameras to enable next-generation intelligent, autonomous systems for AR/VR.

Mr. Li was a recipient of the Best Paper Award at the 2016 IEEE Workshop on Signal Processing Systems.



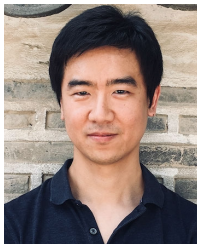
Guanru Wang received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2019, and the M.S. degree in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2021.

Between 2019 and 2021, he was a Graduate Student Research Associate with the Michigan Integrated Circuit Laboratory, Ann Arbor. His research interests include VLSI architectures and algorithm-hardware co-design.



Pierre Abillama received the B.Sc. degree in computer engineering from the University of Minnesota, Twin Cities, MN, USA, in 2020, and the M.Sc. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2022, where he is currently pursuing the Ph.D. degree in electrical engineering.

His current research interests are in low-power and energy-efficient hardware accelerators and processors for deep learning and edge intelligence.



Hun-Seok Kim (Senior Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 2010.

He is currently an Associate Professor with the University of Michigan, Ann Arbor, MI, USA. His research interests focus on system analysis, novel algorithms, and VLSI architectures for low-power/high-performance wireless communications, signal processing, computer vision, and machine-learning systems.

Dr. Kim was a recipient of the DARPA Young Faculty Award in 2018 and the National Science Foundation (NSF) CAREER Award in 2019. He is an Associate Editor of IEEE TRANSACTIONS ON MOBILE COMPUTING.

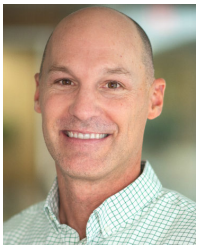
Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.



David Blaauw (Fellow, IEEE) received the B.S. degree in physics and computer science from Duke University, Durham, NC, USA, in 1986, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1991.

Until August 2001, he worked for Motorola Inc., Austin, TX, USA, where he was the Manager of the High Performance Design Technology Group. Since August 2001, he has been on the faculty of the University of Michigan, Ann Arbor, MI, USA, where he is the Kensall D. Wise Collegiate Professor of electrical engineering and computer science (EECS). He has published over 600 articles and holds 65 patents. He has researched ultra-low-power wireless sensors using subthreshold operation and low-power analog circuit techniques for millimeter systems. His research group introduced so-called near-threshold computing, which has become a common concept in semiconductor design. Most recently, he has pursued research in cognitive computing using analog, in-memory neural-networks for edge-devices and genomics for precision health.

Dr. Blaauw was a member of the IEEE International Solid-State Circuits Conference (ISSCC) Analog Program Subcommittee. He received the 2016 SIA-SRC Faculty Award for lifetime research contributions to the U.S. semiconductor industry, the Motorola Innovation Award, and numerous best paper awards. He was awarded the MIT Technology Review's "One of the Year's Most Significant Innovations" for his research. He was the General Chair of the IEEE International Symposium on Low Power.



Dennis Sylvester (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of California at Berkeley (UC Berkeley), Berkeley, CA, USA.

He was the Founding Director of the Michigan Integrated Circuits Laboratory (MICL), Ann Arbor, MI, USA, a group of ten faculties and more than 70 graduate students. He has held research staff positions at the Advanced Technology Group, Synopsys, Mountain View, CA, USA, and the Hewlett-Packard Laboratories, Palo Alto, CA, USA. He also held visiting professorships with the National University of Singapore, Singapore, and Nanyang Technological University, Singapore. He co-founded Ambiq, Austin, TX, USA, a fabless semiconductor company developing ultralow-power mixed-signal solutions for compact wireless devices. He is the Edward S. Davidson Collegiate Professor and the Interim Chair of electrical and computer engineering with the University of Michigan, Ann Arbor. He has published over 550 articles along with one book and several book chapters. He holds 51 U.S. patents. His research interests include the design of millimeter-scale computing systems and energy-efficient near threshold computing.

Dr. Sylvester previously served on the administrative committee for the IEEE Solid-State Circuits Society. He is a fellow of the National Academy of Inventors. His Ph.D. dissertation was recognized with the David J. Sakrison Memorial Prize as the most outstanding research at the Electrical Engineering and Computer Science (EECS) Department, UC Berkeley. He received the National Science Foundation (NSF) CAREER Award, the Beatrice Winner Award at the International Solid-State Circuits Conference (ISSCC), the IBM Faculty Award, the SRC Inventor Recognition Award, and 16 best paper awards and nominations. He was named a top-five Contributing Author all time at ISSCC in 2023 and the Most Prolific Author at the IEEE Symposium on Very Large-Scale Integration (VLSI) Circuits. He was awarded the University of Michigan Henry Russel Award for Distinguished Scholarship. He served as an IEEE Solid-State Circuits Society Distinguished Lecturer. He was an Associate Editor of IEEE JOURNAL OF SOLID-STATE CIRCUITS, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, and IEEE TRANSACTIONS ON VERY-LARGE-SCALE INTEGRATION SYSTEMS. He is an Editor-in-Chief of the IEEE JOURNAL OF SOLID-STATE CIRCUITS.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 28,2025 at 18:06:57 UTC from IEEE Xplore. Restrictions apply.