Analog In-Pixel Computing for Energy-Efficient Visual Intelligence at the Edge

By

Zihan Yin

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical & Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2026

Date of final oral examination: March 3rd, 2026

The dissertation is approved by the following members of the Final Oral Committee:

Akhilesh Jaiswal, Assistant Professor, Electrical & Computer Engineering
Umit Ogras, Gene Amdahl Professor, Electrical & Computer Engineering
Joshua San Miguel, Associate Professor, Electrical & Computer Engineering
Joseph Andrews, Alfred Fritz Assistant Professor, Mechanical Engineering

To my friends and family.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Jaiswal, for his guidance, encouragement, and unwavering support throughout my PhD journey. His insight, mentorship, and high standards have greatly shaped my development as a researcher.

I would also like to sincerely thank my committee members for their time, valuable feedback, and support throughout this work. Their comments and perspectives have helped strengthen this dissertation.

I am also grateful to my labmates for the stimulating discussions, collaboration, and supportive research environment they provided during my years in graduate school.

I am also deeply grateful to my friends, including Dinan Bai, Xingjian Zhang, Zirou Jin, Zhongrui Wang, Zhiyuan Gao, Zishun Jin, Peihao Wang, Yuesong Li, Nairong Du, Yuchen Lin, and Meiyi Guang, for their friendship, support, and encouragement throughout my years in graduate school. Their presence made this journey not only more manageable, but also far more meaningful and memorable. I also wish to thank many other friends whose kindness and companionship supported me along the way.

Above all, I would like to express my heartfelt gratitude to my mother for her unwavering love, sacrifice, and support. Her encouragement and belief in me have always been a constant source of strength throughout this journey.

Contents

List of Tables

List of Figures

Abstract

Extreme-edge intelligence requires sensing and inference under severe energy, bandwidth, and latency constraints. In conventional camera-to-processor pipelines, these constraints are dominated not by arithmetic but by data conversion and movement: dense pixel arrays are digitized at the sensor, transmitted off-chip, and then processed by downstream accelerators. This dissertation proposes and develops a Processing-in-Pixel-in-Memory (P^2M) paradigm for energy-efficient intelligence at the extreme edge, in which sensing signals and learned weights are co-located at pixel granularity so that the front-end stages of convolutional neural network (CNN) inference can be executed within the image sensor, ideally within the pixel array itself. By outputting compact feature representations instead of raw frames, the P^2M paradigm reduces analog-to-digital converter (ADC) activity and sensor-to-processor bandwidth while preserving task performance.

The core premise is that early CNN layers act as feature extractors that can be engineered to provide strong compression, producing feature maps that are smaller and lower-precision than raw sensor outputs. Realizing this inside the sensor requires mixed-signal architectures compatible with stringent pixel-area constraints, and it requires algorithm–circuit co-design to maintain accuracy under analog non-idealities and reduced precision. Within this framework, the dissertation demonstrates a sensor-compatible in-pixel compute pipeline that supports multi-bit, multi-channel convolution and integrates key inference operations using standard readout primitives. In the reported high-resolution TinyML workload configuration, P^2M achieves approximately $21\times$ reduction in sensor output bandwidth and up to approximately $11\times$ improvement in end-to-end energy–delay product (EDP) relative to conventional readout-and-process baselines.

To extend the P^2M paradigm toward broader deployment, the dissertation further develops two complementary directions. First, it introduces a Field-Programmable Pixel Convolutional Array (FPCA) that generalizes processing-in-pixel toward a field-reconfigurable

front-end substrate. FPCA enables reconfiguration of key CNN front-end parameters, including kernel size, stride, and output channel count, through a programmable mapping framework, improving adaptability across tasks and operating points without requiring a new sensor design instance.

Second, it develops a robustness-oriented pathway using capacitive transimpedance amplifier (CTIA)-based in-pixel computing (CTIA-IPC), leveraging CTIA pixel characteristics and a time-domain compute mapping supported by sensor-compatible accumulation and digitization mechanisms to improve fidelity under practical sensing constraints. In the reported case-study evaluation, this pathway achieves approximately $12\times$ bandwidth reduction with only $\sim$1.3–2.5% intersection-over-union (IoU) degradation, and reports throughput and energy-efficiency values of $\sim$1.98 giga operations per second (GOPS) and $\sim$3.39 GOPS/W in the evaluated configuration.

Overall, this dissertation establishes the P^2M paradigm as a practical foundation for energy-efficient intelligence at the extreme edge, and demonstrates that sensor-side CNN front-end processing, when co-designed with circuit behavior, readout constraints, and deployment requirements, can substantially reduce data movement while maintaining effective inference performance.

Chapter 1

Introduction

Modern image sensors generate an enormous volume of data. As camera resolution and frame rate continue to increase, the dominant cost in many vision systems is no longer the arithmetic required for inference, but rather the movement and conversion of data: reading analog pixel signals out of a sensor array, converting them to digital values through analog-to-digital converters (ADCs), and streaming the resulting frames to downstream processors for machine learning inference. This conventional separation of sensing and compute imposes fundamental bottlenecks in energy, bandwidth, and latency that become particularly severe in resource-constrained edge settings, where compute and memory budgets are tight, and in sensing regimes where accuracy stability under challenging conditions (e.g., low light and noise) is critical.

A promising direction to overcome these bottlenecks is to bring computation closer to the sensor. Near-sensor approaches integrate a dedicated accelerator close to the camera, sometimes through advanced packaging, reducing but not eliminating sensor-to-processor data transfers. In-sensor approaches place compute circuits within the periphery of the image sensor die, decreasing off-chip transfers but still requiring significant intra-chip movement of high-resolution pixel data from the array to peripheral compute blocks. The most aggressive strategy is in-pixel computing, where computation is performed within or immediately adjacent to the pixels themselves, enabling massively parallel processing and minimizing data movement at the source. However, practical in-pixel computing faces key obstacles: (i) limited pixel area and strict constraints on fill factor and noise, (ii) analog non-idealities and limited precision, (iii) limited support for the full set of operations needed

by modern convolutional neural networks (CNNs), and (iv) lack of programmability and scalability in many proposals.

This dissertation addresses these challenges by developing in-pixel computing architectures and algorithm–circuit co-design methods that embed the front-end stages of CNN inference directly within CMOS image sensor (CIS) pixel arrays. The central premise is that the first few convolutional layers of a CNN often act as feature extractors and can be engineered to provide strong compression, producing output feature maps whose dimensionality and bit depth are significantly reduced compared to raw input frames. If these layers can be executed inside the sensor array, then only compressed feature representations need to be digitized and transmitted, yielding large reductions in ADC cost and communication bandwidth while preserving task accuracy.

1.1 Dissertation Overview and Scope

The technical focus of this dissertation is on processing-in-pixel architectures that implement multi-channel, multi-bit convolution and associated nonlinear operations within the sensor, while accounting for analog circuit behavior and system-level constraints. The contributions are organized around three major thrusts, each advancing the state of in-pixel computing along a different axis:

1. **Feasibility and end-to-end benefit (P^2M).** Demonstrating that practical CNN front-end operations, including multi-bit, multi-channel convolution, batch normalization, and rectified linear unit (ReLU) activation, can be realized within pixels using manufacturable circuit techniques, and that doing so yields substantial reductions in sensor bandwidth ($\sim 21\times$) and improvements in energy-delay product (up to $\sim 11\times$) on a realistic high-resolution TinyML workload.

2. **Field programmability and reconfigurability (FPCA).** Enabling in-pixel convolution fabrics that can be field-reconfigured across key CNN front-end parameters (kernel size, stride, and channel count) without redesigning the sensor. The pixel array is treated as a reconfigurable convolution substrate through switch-matrix mapping and programmable weight/control mechanisms, enabling adaptation across workloads and operating points.

3. **Robustness and fidelity for practical deployment (CTIA-IPC).** Developing in-pixel computing techniques that improve linearity, stability, and accuracy retention under real-world sensing constraints such as nonlinearity, noise sensitivity, limited precision, and low-light operation. A biomedical near-tissue vision workload serves as a demanding case study, demonstrating 12× bandwidth reduction with only ~1.3–2.5% intersection-over-union (IoU) degradation.

Collectively, these contributions demonstrate a progression from establishing feasibility and quantifying system-level gains (P^2M), to enabling field-programmable front-end mapping (FPCA), to improving robustness and fidelity under practical sensing constraints (CTIA-IPC).

1.2 Key Challenges in In-Pixel Computing

Making in-pixel computing practical for modern deep learning pipelines requires addressing several interrelated challenges:

Data conversion and movement costs. Conventional sensors digitize and transmit full-resolution frames, incurring substantial ADC energy and bandwidth overhead. This is especially inefficient when downstream inference ultimately compresses or discards most raw pixel information after early-layer feature extraction. As illustrated in Fig. 1.1, sensor-to-system-on-chip (SoC) communication alone accounts for 69% of the per-pixel front-end energy in a conventional CIS pipeline, and data movement (ADC conversion plus communication) represents 76% of the total front-end cost. By performing early CNN layers in-pixel and transmitting only compressed features, the front-end energy can be reduced by over 5×.

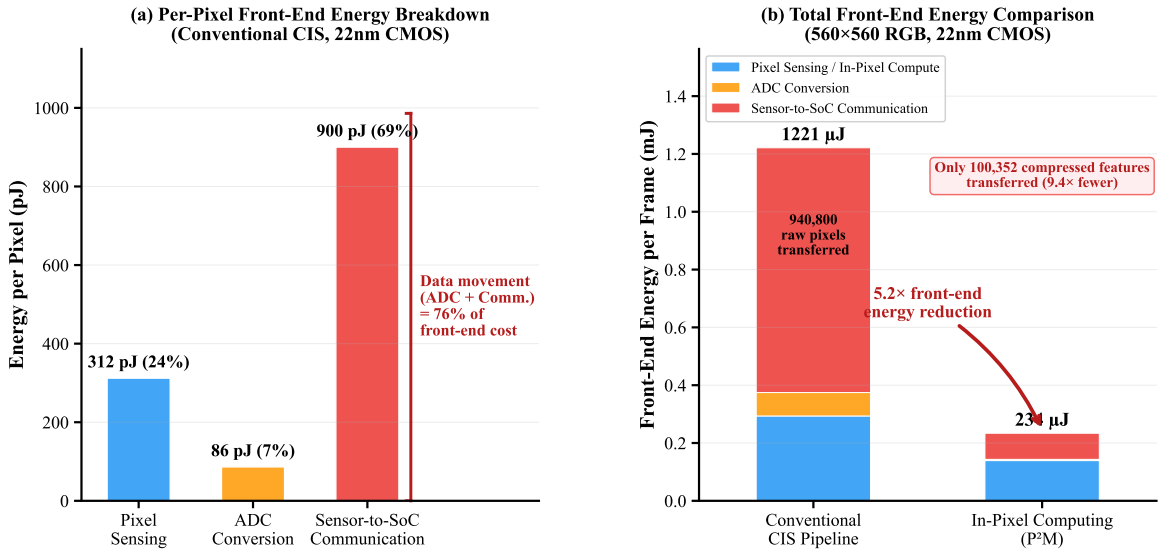


Figure 1.1: Front-end energy breakdown for a conventional CIS pipeline versus in-pixel computing (P^2M) on a 560×560 RGB input using 22nm CMOS technology. (a) Per-pixel energy breakdown showing that data movement (ADC + communication) accounts for 76% of the front-end cost. (b) Total front-end energy comparison: in-pixel computing reduces the number of features requiring ADC conversion and communication from 940,800 raw pixels to 100,352 compressed features, yielding a 5.2× front-end energy reduction.

Limited pixel real estate and manufacturability. Pixels are optimized for photonic efficiency, noise, and array density. Embedding computation and weight storage directly into pixels can inflate transistor count and reduce fill factor, limiting scalability unless the architecture is carefully designed or weight storage is displaced to a 3D-integrated tier.

Analog non-idealities and limited precision. In-pixel computation often leverages analog behavior such as voltage/current accumulation and charge integration. The resulting nonlinearities and mismatch can distort dot products and degrade learning accuracy unless explicitly modeled and compensated within the training and deployment flow.

Support for complete CNN front-end operations. CNN inference requires more than dot products. Practical in-pixel execution needs support for signed weights, batch normalization, and nonlinear activation (e.g., ReLU), and must integrate with conventional sensor readout circuits and timing constraints.

Programmability. Many deployment scenarios require the first-layer parameters to be tunable across networks and tasks. Fixed-function in-pixel layers limit flexibility, motivating architectures that can adapt kernel size, stride, channel count, and weight mappings without redesigning the sensor.

This dissertation targets these issues by combining circuit architecture innovation with algorithm-level adaptation, so that the network is designed jointly with the sensor compute substrate.

1.3 Contributions of This Dissertation

This dissertation makes the following primary contributions:

1.3.1 Processing-in-Pixel-in-Memory (P^2M)

The first contribution introduces a Processing-in-Pixel-in-Memory (P^2M) paradigm that co-locates input activations (photodiode-generated signals) and multi-bit weights within pixel structures to implement massively parallel, multi-channel analog convolution. To support signed weights and nonlinear functions, the approach repurposes existing sensor circuits, specifically correlated double sampling (CDS) and single-slope ADC, to realize positive/negative accumulation, batch normalization through counter initialization, and quantized ReLU activation. A key component is an algorithm–circuit co-design methodology that models pixel-level analog non-idealities (captured through SPICE-calibrated curve-fitting functions) during CNN training, so that the network learns to compensate for transistor-level nonlinearities. On the Visual Wake Words dataset at 560×560 resolution, P^2M reduces sensor-to-processor bandwidth by $\sim21\times$ and improves the overall energy-delay product by up to $\sim11\times$, while maintaining classification accuracy within 1.47% of the uncompressed baseline.

1.3.2 Field-Programmable Pixel Convolutional Array (FPCA)

The second contribution advances in-pixel computing from a fixed-function front-end toward a field-programmable substrate. FPCA introduces a compact 4T pixel architecture with hybrid CMOS–non-volatile memory (NVM) weight storage that enables post-fabrication

reconfiguration of kernel size, stride, output channel count, and weight values through programmable switch-matrix routing. This reconfigurability allows the same in-pixel compute fabric to adapt to different network front-ends and bandwidth-reduction targets rather than being tailored to a single architecture instance. FPCA also supports practical deployment features such as flexible spatial mapping of convolution windows, region skipping for efficiency, and a novel bucket-select curve-fit modeling approach that accurately captures the analog non-idealities of the circuit for integration into machine learning training frameworks. Energy, latency, and bandwidth-reduction analyses on the TSMC 28nm HPC+ technology node demonstrate the dependence of these metrics on CNN hyperparameters, highlighting the importance of algorithm–hardware co-design.

1.3.3 CTIA-Based In-Pixel Computing (CTIA-IPC)

The third contribution focuses on improving the robustness and fidelity of in-pixel computing under practical sensing conditions where non-idealities can significantly degrade dot-product accuracy and downstream model performance. The dissertation develops an in-pixel computing pathway based on Capacitive Transimpedance Amplifier (CTIA) pixel structures, which offer superior linearity and low-noise performance compared to conventional 3T/4T active pixel sensors. The approach implements multiply–accumulate operations using a weight-to-time conversion scheme with 3D-stacked static random-access memory (SRAM) for per-pixel weight storage, while reusing sensor ADC and digital CDS circuitry for signed computation, batch normalization, and 4-bit ReLU activation. Using the EndoVis surgical segmentation benchmark as a demanding biomedical case study, CTIA-IPC achieves $12\times$ bandwidth reduction with only $\sim$1.3–2.5% IoU degradation relative to a software baseline, while delivering a throughput of 1.98 giga operations per second (GOPS) at an energy efficiency of 3.39 GOPS/W.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 reviews background on CMOS image sensors, in-pixel and in-sensor computing paradigms, and the algorithm–hardware considerations needed to integrate CNN front-end operations within sensor arrays. Chapter 3 presents the Processing-in-Pixel-in-Memory (P^2M) paradigm and its

algorithm–circuit co-design methodology. Chapter 4 introduces the Field-Programmable Pixel Convolutional Array (FPCA) and its reconfigurable mapping approach. Chapter 5 presents the CTIA-based in-pixel computing pathway, emphasizing robustness and fidelity under practical sensing constraints and validating accuracy–bandwidth trade-offs on a biomedical case study. Chapter 6 describes the silicon prototype of the P^2M architecture, including chip fabrication in GlobalFoundries 22nm FD-SOI, test board design, and the ongoing measurement campaign. Finally, Chapter 7 summarizes the dissertation contributions and discusses future directions.

Chapter 2

General Background

This chapter provides the common background knowledge that underpins all subsequent technical chapters. We begin with the fundamentals of CMOS image sensor (CIS) readout and the mixed-signal primitives that this dissertation repurposes for computation. We then categorize compute-placement approaches (near-sensor, in-sensor, and in-pixel) and examine the data-movement bottleneck that motivates in-pixel processing, followed by the convolutional neural network (CNN) front-end operations targeted by our architectures. Finally, we define the system-level metrics used throughout the dissertation, discuss hardware non-idealities and the algorithm–circuit co-design philosophy, and position the contributions of this dissertation relative to prior work.

2.1 CMOS Image Sensor Fundamentals and Readout Primitives

CMOS image sensors (CIS) convert incident light into electrical signals using a two-dimensional array of pixels, followed by column-level and peripheral readout circuits that condition, digitize, and transmit the data [3]. While conventional CIS are designed primarily for image formation, several building blocks in standard readout pipelines, especially correlated double sampling (CDS) and single-slope analog-to-digital conversion (SS-ADC), can also be viewed as mixed-signal primitives that support computation near or within the pixel array. This dissertation leverages these primitives as foundational "in-sensor resources" to implement parts of convolutional neural network (CNN) front-end inference with minimal disruption

to manufacturable CIS pipelines.

2.1.1 Pixel Array and Column-Parallel Readout

A CIS pixel typically contains a photodetector (photodiode) and a small set of transistors that enable reset, signal conversion, and readout [3]. Pixels are arranged in a two-dimensional array, and the readout is commonly organized in a column-parallel fashion: each column shares analog circuitry (e.g., sample-and-hold, comparator) and often an ADC, while rows are selected sequentially in a rolling-shutter manner or in groups depending on the sensing mode [4]. The essential system constraint is that, in conventional operation, the pixel array produces a dense stream of pixel values that must be digitized and moved off-array, making ADC energy and data transfer bandwidth key bottlenecks for downstream processing.

2.1.2 APS versus CTIA Pixels

Two pixel families are particularly relevant to this dissertation: active pixel sensor (APS)-style readout and capacitive transimpedance amplifier (CTIA) pixels.

APS-style pixels. APS pixels are widely used and typically read out a voltage that depends on the photodiode signal and source-follower behavior [3]. These structures are compact and array-friendly, making them natural starting points for embedding additional functionality in the pixel array. In the P^2M and FPCA paradigms (Chapters 3 and 4), APS-style operation provides the baseline sensing functionality while enabling additional in-pixel behaviors via weight modulation and array-level accumulation.

CTIA pixels. CTIA pixels integrate photocurrent onto a feedback capacitor through an operational transconductance amplifier (OTA), producing an output voltage

$$V_{out} = \frac{Q_{photo}}{C_f}, \tag{2.1}$$

where Q_{photo} is the collected photocharge and C_f is the feedback capacitance. This relationship is often more linear with respect to collected charge over the integration interval compared to APS readout [5, 6]. The linearity and stability of CTIA pixels can be especially

valuable in regimes where fidelity under low-signal conditions is important. In this dissertation, CTIA pixels serve as a pathway to robust in-pixel computing (Chapter 5), where maintaining multiply–accumulate (MAC) accuracy under practical sensing constraints (e.g., low light, noise, and process non-idealities) becomes a first-class design objective.

The APS vs. CTIA distinction is emphasized here not to survey pixel design exhaustively, but because it motivates why certain compute mappings are more naturally supported, or more robust, on one pixel class than the other in later chapters.

2.1.3 Correlated Double Sampling as a Differencing Primitive

A key challenge in mapping CNN computations into the sensor is the need to support signed accumulation (positive and negative weights) while operating with physically non-negative device parameters (e.g., transistor widths, currents, integration times). Modern CIS often employ correlated double sampling (CDS) to reduce fixed-pattern noise and reset noise by capturing two correlated samples and subtracting them [7, 8]. In a typical pipeline, one sample corresponds to a reset or reference level (V_{reset}) and the second corresponds to the signal level (V_{sig}); the CDS output is

$$V_{\text{CDS}} = V_{\text{sig}} - V_{\text{reset}}, \tag{2.2}$$

which suppresses offset-like noise components.

This dissertation leverages the differencing behavior of CDS as a computational primitive. Instead of using CDS solely for noise cancellation, the same two-sample-subtract mechanism can compute the difference of two accumulated quantities, naturally supporting positive and negative contributions. Specifically, one phase accumulates the dot-product output for positive weights (counter counts up) and a second phase accumulates the output for negative weights (counter counts down), so that the final digitized value represents a signed inner product. This idea is used to enable signed dot products in both P^2M-style and CTIA-based in-pixel computing architectures developed in Chapters 3–5.

2.1.4 Single-Slope ADC and Counter-Based Digitization

In many CIS architectures, especially those emphasizing column-parallel simplicity and scalability, single-slope ADCs (SS-ADCs) are used for digitization [4, 7]. An SS-ADC com-

pares the input analog voltage against a linearly ramping reference signal. A counter runs while the ramp progresses; when a comparator toggles, the latched count encodes the input voltage. SS-ADCs are attractive because they map naturally to column-parallel layouts and can share the ramp generator across all columns.

From a compute perspective, SS-ADCs provide two useful properties:

1. **Existing infrastructure.** SS-ADCs already reside in many CIS readout pipelines, meaning that repurposing their behavior avoids adding large dedicated compute blocks in the sensor periphery.

2. **Manipulable counter state.** If one phase causes the counter to count up and another phase causes it to count down, the final count directly represents a difference of two digitized quantities, aligning with the CDS-based signed accumulation described above. Moreover, clipping behaviors (for example, preventing negative final counts) can implement simple nonlinearities such as a quantized ReLU-like operation under appropriate counter control logic.

These properties are used as part of the in-sensor realization of signed accumulation and activation in later chapters.

2.1.5 Why These Primitives Are Central to In-Pixel CNN Front-Ends

The in-pixel CNN front-end mappings developed in this dissertation rely on a consistent design strategy: keep the pixel array highly parallel, perform early compressive operations (e.g., first-layer convolution and lightweight nonlinear processing), and transmit only a reduced feature representation to downstream compute. Achieving this within manufacturable CIS constraints is made substantially easier by treating CDS and SS-ADC not merely as sensing utilities, but as reusable mixed-signal compute primitives. This framing reduces architectural disruption, preserves compatibility with column-parallel readout structures, and provides a path to implement essential CNN behaviors, including signed accumulation, normalization and offset handling, and simple activation, within the sensor data path.

2.2 Categories of Compute Placement and the Data-Movement Bottleneck

To understand why in-pixel computing is compelling, and where it fits relative to other approaches, it is useful to organize sensor-centric intelligence solutions by where computation is placed relative to the pixel array. This section describes three categories: near-sensor, in-sensor, and in-pixel processing (illustrated in Fig. 2.1), and explains why data movement and conversion often dominate system cost in conventional pipelines.

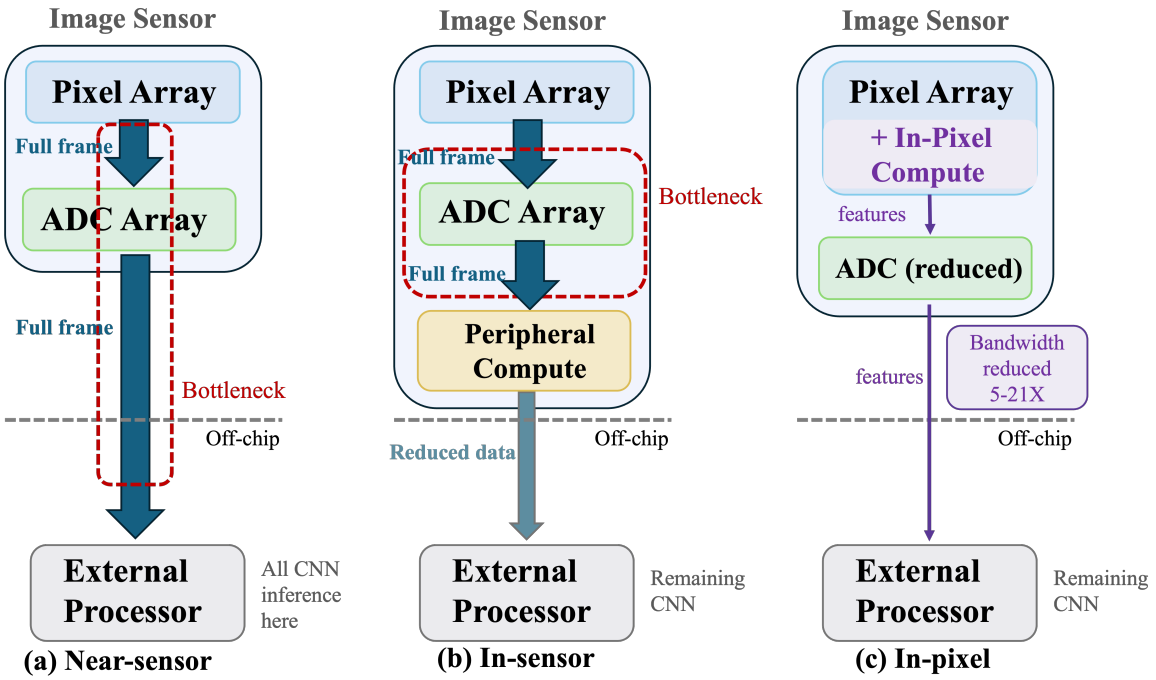


Figure 2.1: Comparison of data movement in three compute-placement approaches. (a) Near-sensor processing: the full-resolution frame is digitized and transferred off-chip to an external processor. (b) In-sensor processing: computation is performed in the sensor periphery after array readout, reducing off-chip transfer but retaining intra-chip data movement. (c) In-pixel processing: computation occurs within the pixel array itself, so only a compressed feature representation needs to be digitized and transmitted. Arrow thickness indicates relative data volume; red dashed outlines highlight the dominant data-movement bottleneck in each approach.

2.2.1 Near-Sensor Processing

In near-sensor processing, the image sensor remains largely conventional: it captures frames, digitizes them via column-parallel ADCs, and transfers the full digital output to a nearby processor (e.g., microcontroller, CPU/GPU, or a dedicated accelerator) located on the same board or in the same package [9, 10]. This approach reduces the distance data travels compared to cloud-based processing and can improve privacy and latency. However, it still requires transferring high-volume digitized image data away from the sensor, so sensor ADC energy and sensor-to-processor bandwidth remain major bottlenecks, especially for high-resolution cameras and continuous inference workloads [11].

2.2.2 In-Sensor Processing

In in-sensor processing, some computation is integrated into the sensor die, typically in the periphery outside the pixel array [12]. By moving compute onto the sensor chip, this approach can reduce off-chip bandwidth and enable tighter coupling between sensing and processing. However, many in-sensor designs still require pixel data to be read out of the array and delivered to peripheral circuits through column buses or array readout paths. As a result, while off-chip transfer may be reduced, the architecture still incurs the cost of moving large amounts of data from the pixel array to the periphery, often after full or partial ADC conversion [12].

2.2.3 In-Pixel Processing

In in-pixel processing, computation is pushed into the pixel array itself, or immediately adjacent to it at pixel granularity [13–15]. This is the most aggressive approach for minimizing data movement because the earliest processing stages occur where the signal is generated. In-pixel processing can exploit massive parallelism across the array, potentially reducing both the amount of data that must be digitized and the volume that must be transmitted downstream. The key challenge is that pixel area, noise, and manufacturability constraints are strict; therefore, architectures must reuse existing sensor circuitry when possible and avoid excessive per-pixel overhead.

2.2.4 Why Data Movement Dominates

In conventional camera-to-processor pipelines, each pixel is digitized (often at 10–12 bits per pixel) and transmitted as part of a full-resolution frame. For modern resolutions and frame rates, this produces a large stream of data that must pass through ADCs and communication links [16]. Importantly, in many deep learning systems, the earliest CNN layers immediately transform and compress these inputs into feature maps whose spatial dimensions and effective precision can be much smaller than the raw image. This observation motivates a key idea explored throughout this dissertation: *if a portion of these front-end CNN operations can be performed within the sensor, ideally within the pixel array, then the sensor can output a compressed feature representation rather than raw frames.*

The P^2M paradigm (Chapter 3) demonstrates this principle concretely by mapping multi-bit, multi-channel convolution and associated operations into the pixel array, leading to substantial reduction in the amount of data requiring ADC conversion and transmission, on the order of $\sim 21\times$ reduction in sensor output bandwidth in the evaluated configuration.

2.2.5 Implications for Architecture Design

This compute-placement viewpoint highlights a core trade-off:

- **Near-sensor and in-sensor** approaches can reduce or eliminate cloud communication, but may still incur large sensor readout and ADC costs because raw or minimally processed pixel data must leave the array.
- **In-pixel** approaches have the potential to reduce both ADC activity and data bandwidth by performing early compressive operations before readout, but must be designed around pixel-area constraints and analog non-idealities.

The architectures developed in this dissertation aim to capture the benefits of in-pixel processing while addressing these constraints through (i) embedding weights at pixel granularity to enable fixed in-situ convolution (P^2M, Chapter 3), (ii) enabling field-reconfigurable convolution mapping in the pixel array (FPCA, Chapter 4), and (iii) improving robustness and fidelity under practical sensing constraints using CTIA-based in-pixel computing (CTIA-IPC, Chapter 5).

2.3 CNN Front-End Operations Targeted by In-Pixel Computing

This dissertation focuses on executing the front-end of convolutional neural networks (CNNs) [17] within the image sensor, with emphasis on operations that (i) dominate early inference cost at high resolution, and (ii) provide strong leverage for data reduction before readout. In particular, the first one or two convolutional stages often transform raw pixels into feature maps that are both more compact and more task-relevant than the original image [18]. By implementing these stages inside the sensor, ideally inside the pixel array, the sensor can transmit compressed feature representations rather than full-resolution frames.

2.3.1 Convolution and Dot-Product Structure

A two-dimensional convolution layer computes output feature maps by sliding a kernel across the input image (or input feature maps) and computing a weighted sum at each position [17, 19]. For an input with C_{in} channels and an output with C_{out} channels, each output activation is a dot product between a local receptive field and a learned weight tensor. While later CNN layers often operate on reduced spatial dimensions, the first layer operates on the full sensor resolution (or close to it), making the first-layer multiply-and-accumulate (MAC) count and data movement particularly expensive in conventional pipelines.

In the architectures studied in this dissertation, convolution is mapped into the sensor using massively parallel operations distributed across the pixel array and column structures. Two key implementation requirements follow directly from the CNN convolution structure:

- **Multi-channel support.** Practical CNN front-ends require accumulation across multiple input channels (e.g., three RGB channels) or multiple output feature maps. This motivates architectures that can support multi-channel accumulation without excessive per-pixel overhead.
- **Multi-bit weights and activations.** To maintain accuracy on real-world tasks, the mapping should support multi-bit values or an effective multi-bit representation through spatial encoding (e.g., transistor width) or temporal encoding.

2.3.2 Signed Weights and Positive/Negative Accumulation

CNN weights are typically signed [20, 21]. However, many in-sensor and in-pixel circuits are naturally constrained to non-negative physical quantities such as currents, charges, integration times, or transistor conductances. A common strategy is therefore to represent a signed dot product as the difference of two non-negative accumulations:

1. accumulate contributions from positive weights in one phase, and
2. accumulate contributions from negative weights in another phase, then subtract.

This dissertation adopts that principle and realizes it using sensor readout primitives, specifically CDS-based differencing and SS-ADC counter manipulation (described in Section 2.1), enabling signed accumulation without requiring a full signed analog datapath inside each pixel.

2.3.3 Stride, Receptive Field, and Output Feature Map Sizing

Three CNN front-end hyperparameters are particularly important when mapping convolution into the sensor:

- **Kernel size** (receptive field): e.g., 3×3, 5×5, or 7×7. Larger kernels increase compute and routing complexity but can improve early feature extraction in some model families [20, 22].
- **Stride**: the step size of the sliding window. Increasing stride reduces the output spatial resolution and can provide strong bandwidth savings, but may reduce accuracy if too aggressive [21].
- **Channel count**: the number of output channels determines representational capacity and affects both compute cost and output bandwidth.

Because these parameters shape both the computational workload and the amount of output data produced, they also determine the achievable bandwidth reduction when early layers are executed inside the sensor. The FPCA architecture (Chapter 4) specifically targets practical deployment by enabling field reconfiguration of kernel size, stride, and channel count in the pixel-array convolution mapping, allowing the in-pixel front-end to be adapted across tasks and operating points.

2.3.4 Batch Normalization and Activation

In modern CNNs, a convolution is typically followed by batch normalization (BN) [23] and a nonlinear activation such as the rectified linear unit (ReLU) [24]. At inference time, BN reduces to an affine transformation:

$$y = \alpha x + \beta, \tag{2.3}$$

where $\alpha = \gamma/\sqrt{\sigma^2 + \epsilon}$ and $\beta = \beta_{BN} - \gamma\mu/\sqrt{\sigma^2 + \epsilon}$ are constants derived from training statistics (γ, β_{BN}, μ, σ). This affine structure makes BN particularly amenable to hardware implementation: the scale factor α can be absorbed into the convolution weights (weight scaling), and the offset β can be implemented as an initialization value in the accumulation or digitization path.

The ReLU activation clips negative values to zero:

$$\mathrm{ReLU}(x) = \max(0, x). \tag{2.4}$$

In the counter-based SS-ADC readout described in Section 2.1.4, this clipping can be realized by preventing the counter from latching negative final values after the CDS subtraction phase.

In this dissertation, BN and ReLU are treated as part of the sensor front-end mapping rather than as separate downstream steps, and are implemented by reusing readout structures such as SS-ADC counters and CDS initialization behaviors.

2.3.5 Why the Dissertation Targets the Front-End

The design choices described above are motivated by a consistent system objective: reduce the amount of data that must be digitized and transmitted by applying compressive computation at the earliest possible point in the pipeline. The first convolutional layer of modern compact architectures such as MobileNetV2 [21] can be configured with large non-overlapping strides and a reduced channel count to achieve strong spatial compression while preserving task-relevant features [18]. Executing this front-end convolution, along with lightweight post-processing (BN and ReLU), inside the sensor can reduce output bandwidth substantially. For example, the P^2M mapping evaluated in Chapter 3 reports

$\sim$21$\times$ reduction in sensor output bandwidth, while the CTIA-IPC architecture in Chapter 5 demonstrates $\sim$12$\times$ bandwidth reduction while maintaining strong task accuracy on a surgical segmentation workload.

2.4 Bandwidth Reduction and System Metrics

Because the primary goal of in-pixel CNN front-ends is to reduce data conversion and movement, this dissertation uses a small set of system-level metrics consistently across chapters. The most important metric is bandwidth reduction (BR), which quantifies how much less data must be digitized and transmitted compared to streaming raw pixel outputs. Energy and latency metrics, particularly energy-delay product (EDP), are then used to capture end-to-end system benefit.

2.4.1 Bandwidth Reduction

Definition. Let I denote the number of elements in the raw input image representation (e.g., RGB pixel values), O the number of elements in the output activation map produced by the in-pixel front-end layer(s), and N_b the bit precision of the output activations. BR is then defined as:

$$BR = \frac{I}{O} \times \frac{4}{3} \times \frac{12}{N_b}. \tag{2.5}$$

Interpretation of factors. The three multiplicative terms in Eq. 2.5 each capture a distinct source of compression:

- I/O captures how much the network front-end reduces the number of data elements, primarily through strided convolution and output channel sizing.
- $4/3$ accounts for converting the Bayer RGGB mosaic pattern produced by the sensor into an effective RGB representation. The raw sensor produces four sub-pixels (RGGB) for every three color channels; this factor corrects for that conversion.
- $12/N_b$ accounts for reducing bit precision from the typical raw sensor depth (often 12 bits per pixel in modern CIS [25]) to the lower activation precision N_b used by the quantized front-end output.

Output size. For a convolutional layer with square input spatial dimension i, kernel size k, padding p, stride s, and c_o output channels, the output element count is:

$$O = \left(\frac{i-k+2p}{s}+1\right)^2 \times c_o, \qquad I = i^2 \times 3. \tag{2.6}$$

All three architectures in this dissertation (P^2M, FPCA, and CTIA-IPC) use BR with the same structure, differing only in the specific values of stride, kernel size, channel count, and output precision dictated by each design. For instance, P^2M achieves a BR of $\sim 21\times$ using a 5×5 kernel with stride 5 and 8 output channels at 8-bit precision, while CTIA-IPC reports a BR of $\sim 12\times$ using a 7×7 kernel with stride 2 and 16 output channels at 4-bit precision.

Practical note. Throughout this dissertation, BR serves as a system proxy for how much ADC activity and I/O bandwidth can be avoided by outputting compressed feature maps rather than full frames.

2.4.2 Energy, Latency, and Energy-Delay Product

To compare conventional pipelines against in-pixel front-end execution, total inference energy is decomposed into major components, and energy-delay product (EDP) is used as a compact joint metric.

Energy breakdown. Total energy is modeled as the sum of four components:

$$E_{tot} \approx \underbrace{(e_{pix}+e_{adc})\cdot N_{pix}}_{E_{sens}} + \underbrace{e_{com}\cdot N_{pix}}_{E_{com}} + \underbrace{e_{mac}\cdot N_{mac}}_{E_{mac}} + \underbrace{e_{read}\cdot N_{read}}_{E_{read}}, \tag{2.7}$$

where e_{pix} and e_{adc} are per-pixel sensing and ADC energy, e_{com} is per-pixel sensor-to-SoC communication energy, e_{mac} is the energy per MAC operation, and N_{pix}, N_{mac}, N_{read} are the respective operation counts.

This decomposition is useful for two reasons. First, it makes explicit that reducing N_{pix} (via BR) directly attacks both ADC cost and I/O cost, which together dominate the front-end energy budget (see Fig. 1.1). Second, it separates sensor-side savings from downstream compute cost, allowing fair comparisons across compute-placement strategies.

Latency. When discussing sensor-side front-end compute, "latency" refers to the time required to produce the entire output feature map for the in-pixel layer(s), including exposure, readout cycles, and ADC conversion. For sequential layer execution, the total delay can be expressed as:

$$T_{delay} \approx T_{sens} + T_{adc} + T_{conv}, \tag{2.8}$$

where T_{sens} and T_{adc} correspond to sensor read and ADC operation delays, and T_{conv} captures the delay of all downstream convolutional layers.

Energy-delay product (EDP). EDP is used to capture the combined benefit of lowering energy while maintaining or improving throughput. In this dissertation, EDP comparisons quantify the system-level advantage of executing CNN front-end layers inside the sensor rather than a single component in isolation. P^2M (Chapter 3) reports up to $\sim 11\times$ EDP improvement in its evaluated configuration, alongside its $\sim 21\times$ bandwidth reduction.

2.4.3 Throughput and Energy Efficiency

For architectures that explicitly model sensor-side MAC throughput and efficiency, two additional metrics are reported:

- **Throughput** (GOPS): total operations per second supported by the sensor-side compute pipeline.
- **Energy efficiency** (GOPS/W): throughput normalized by power consumption.

These metrics are most relevant for the CTIA-IPC architecture (Chapter 5), which reports $\sim$1.98 GOPS throughput and $\sim$3.39 GOPS/W energy efficiency (including pixel readout and digital CDS energy), alongside a BR of $\sim 12\times$.

2.5 Hardware Non-Idealities and Algorithm–Circuit Co-Design Philosophy

A central challenge in in-pixel and in-sensor neural processing is that the compute substrate is not an ideal digital MAC engine. Instead, it is a mixed-signal system built from devices

and readout circuits originally optimized for sensing. As a result, the physical implementation introduces non-idealities, including nonlinearity, mismatch, noise, finite precision, and limited dynamic range, that can distort dot products and degrade end-task accuracy if they are not explicitly accounted for [26, 27]. This dissertation adopts an algorithm–circuit co-design approach in which (i) circuit-level behavior is characterized or modeled, (ii) that behavior is integrated into training and inference-time models, and (iii) network and mapping parameters are chosen to maximize system-level benefit (e.g., bandwidth reduction) while maintaining accuracy.

2.5.1 Sources of Non-Idealities in Sensor-Side Compute

Non-idealities arise at multiple levels of the sensor compute pipeline:

Device and pixel-level non-idealities. Pixel circuits operate under tight area and power budgets, often relying on transistor operating regions and capacitive integration behavior. This can introduce nonlinear transfer characteristics, limited voltage swing, and sensitivity to process variation. In architectures that perform accumulation or multiplication using analog behaviors (e.g., current integration or source-follower modulation), these effects can directly distort the intended MAC relationship.

Array-level effects and mismatch. When computations are distributed across large arrays, device mismatch and spatial variation can appear as fixed-pattern offsets or gain variation across columns and rows. While conventional imaging pipelines mitigate some of these effects via CDS (Section 2.1.3), compute mappings can stress the array in different operating modes, making residual mismatch more visible in the computed outputs.

Readout and digitization non-idealities. Column-level circuits (comparators, ramp generators, counters) introduce quantization effects, timing granularity, and potential systematic distortions. Since in-pixel computing often aims to output lower-bit activations to reduce bandwidth, quantization becomes an integral part of the compute model rather than a negligible back-end effect.

Mapping constraints and limited programmability. Practical constraints such as limited per-pixel storage, restricted kernel shapes, stride implementation, and routing

patterns can force approximations of ideal CNN operations. These "structural constraints" represent non-idealities at the architectural level: even if a circuit is accurate, the mapping itself may deviate from a fully flexible digital layer. The FPCA architecture (Chapter 4) treats this as a first-class design axis by enabling field reconfiguration of kernel size, stride, and channel count, but the mapping still requires careful coordination between the network configuration and the hardware schedule.

2.5.2 Why Circuit-Aware Modeling Is Necessary

If hardware behavior deviates from an ideal dot product, then applying a network trained under ideal arithmetic can lead to accuracy loss. This is especially true when bandwidth reduction forces aggressive quantization and when analog distortions accumulate over spatial windows involving tens or hundreds of pixels. Many existing works in the domain of memristive analog computing ignore non-idealities arising from nonlinear transistor behavior [26, 27]. In contrast, the key principle used throughout this dissertation is:

> *Train and evaluate the network under a model that reflects the hardware transfer function.*

In practice, this includes:

- modeling nonlinear input to output behavior with fitted functions derived from circuit simulation,
- including quantization and clipping behavior consistent with the readout path, and
- enforcing the same structural constraints (kernel size, stride, channel count, and mapping schedule) used by the sensor-side implementation.

In P^2M (Chapter 3), this method appears as an explicit algorithm–circuit co-design methodology where pixel-level non-idealities, captured through SPICE-calibrated curve-fitting functions, are incorporated into the PyTorch training pipeline so that the network learns to compensate for transistor-level nonlinearities. In FPCA (Chapter 4), the modeling perspective is similarly emphasized to make the field-programmable mapping compatible with training and evaluation under hardware-realistic behavior.

2.5.3 Co-Design Workflow

Across the three architectures developed in this dissertation, the co-design workflow can be summarized in the following steps:

1. **Define the target front-end operations and system objective.** Select the front-end layer(s) to execute in-sensor (e.g., first convolution plus batch normalization and ReLU) and define the primary objective, such as bandwidth reduction, energy efficiency, or robustness under sensing constraints.

2. **Establish the hardware mapping and constraints.** Determine how convolution windows are realized (spatial mapping), how signed weights are supported (e.g., positive/negative CDS phases), and what bit precision and readout schedule are feasible. For FPCA, this includes the field-reconfigurable parameters (kernel size, stride, channel count) and routing schedule constraints.

3. **Characterize or model the compute transfer behavior.** Use circuit simulation (e.g., SPECTRE on the target technology node) and fitted models to capture the effective input–output relationship, including nonlinearity, quantization, and clipping behaviors introduced by the sensor readout chain.

4. **Train and evaluate with hardware-aware models.** Integrate the modeled behavior into the training or calibration pipeline so that the learned parameters compensate for systematic distortions and operate reliably under reduced-precision outputs.

5. **Validate system-level trade-offs.** Evaluate the final design on task-level metrics (classification accuracy, segmentation IoU) alongside system metrics (BR, EDP, throughput and efficiency). The CTIA-IPC architecture (Chapter 5), for example, evaluates robustness using a surgical segmentation case study under aggressive bandwidth reduction to quantify accuracy stability under practical sensing constraints.

2.5.4 Robustness as a First-Class Co-Design Goal

In addition to achieving bandwidth and energy benefits, practical use requires that sensor-side computation remain stable as sensing conditions vary. Low-light operation, noise sensitivity, and linearity limitations can change the effective compute behavior and degrade

inference quality. The CTIA-based pathway developed in this dissertation (Chapter 5) is motivated in part by improving the fidelity of the compute primitive itself. By leveraging the inherent linearity of CTIA pixel integration (Section 2.1.2) and coupling it with weight-to-time conversion and reuse of on-chip readout mechanisms, the CTIA-IPC architecture aims to preserve MAC accuracy and downstream task performance under challenging sensing regimes.

2.5.5 Summary

Non-idealities are not an implementation detail but a defining feature of sensor-side computing. The dissertation's approach is to embrace this reality through algorithm–circuit co-design: mapping CNN front-end operations into the sensor in a way that is compatible with manufacturable readout pipelines, while explicitly modeling and compensating hardware behavior to maintain accuracy under aggressive bandwidth reduction and practical sensing constraints.

2.6 Related Work in Sensor-Centric Computing

Processing-in-pixel and near/in-sensor computing have been explored as ways to reduce the bandwidth, energy, and latency costs of transferring raw high-resolution sensor data to downstream processors. This section provides a brief overview of prior architectural directions and identifies the gaps that motivate the contributions of this dissertation. A more detailed discussion of related work specific to each architecture appears in Chapters 3–5.

Digital pixel-processor arrays. One class of approaches implements pixel-parallel digital processing using single instruction, multiple data (SIMD)-like pixel processor arrays. These designs offer programmability and deterministic execution, and can embed certain neural operators directly into pixel-addressable compute substrates. However, because computation proceeds through digital instruction sequences with strict per-pixel resource budgets, such systems can suffer reduced effective parallelism relative to analog accumulation for dot-product-heavy operations. Representative examples include the work of Bose et al. [14], which embeds convolutional networks on a pixel processor array but is limited to fully connected layers and simple datasets such as MNIST.

Mixed-signal and computational image sensors. A second direction integrates mixed-signal processing into the sensor readout chain to perform feature extraction and reduce output bandwidth. Examples include programmable convolutional imager SoCs with current-domain ternary-weighted MAC operations [28], computational CMOS image sensors with programmable kernels for feature extraction [29], and reconfigurable convolution-in-pixel architectures that represent weights through pixel exposure time [15, 30]. In-sensor binary or analog convolution schemes have also been proposed using peripheral processing [12] or emerging 2D materials [13]. Despite their promise, many of these approaches operate under constraints that limit end-to-end CNN deployment at the focal plane: restricted weight precision (e.g., ternary or binary), limited channel scalability, or incomplete support for common inference operators such as batch normalization and ReLU when viewed as a full front-end pipeline.

Modeling tools for in-pixel accelerators. Beyond hardware architectures, behavior-level modeling tools such as PiPSim [31] have been proposed to evaluate processing-in-pixel accelerators and study how analog non-idealities affect inference outcomes. This complementary research direction reinforces the importance of algorithm–hardware co-design when analog behavior materially affects accuracy and robustness.

Gap summary and thesis positioning. Taken together, prior work motivates the focal-plane compute direction but leaves several gaps for practical CNN front-end deployment inside the pixel array:

1. **Full front-end capability:** support for multi-bit, multi-channel convolution with integration of batch normalization and nonlinear activation, rather than a partial compute primitive alone.

2. **Scalability and flexibility:** mechanisms that adapt kernel size, stride, and channel count without requiring a new sensor design, enabling broader deployment across tasks and operating points.

3. **Robustness under real sensor physics:** maintaining accuracy when non-idealities, nonlinearity, and low-light operation significantly affect dot-product fidelity and downstream inference.

This dissertation addresses these gaps through a progression of contributions: (i) demonstrating end-to-end feasibility and system benefit for in-pixel CNN front-ends (P^2M, Chapter 3), (ii) introducing field programmability for practical deployment adaptability (FPCA, Chapter 4), and (iii) advancing robustness and fidelity through CTIA-based circuit design and co-design methodology (CTIA-IPC, Chapter 5).

Chapter 3

A processing-in-pixel-in-memory paradigm for resource-constrained TinyML applications (P2M)

3.1 Introduction

Today's widespread applications of computer vision spanning surveillance [32], disaster management [33], camera traps for wildlife monitoring [34], autonomous driving, smartphones, etc., are fueled by the remarkable technological advances in image sensing platforms [35] and the ever-improving field of deep learning algorithms [36]. However, hardware implementations of vision sensing and vision processing platforms have traditionally been physically segregated. For example, current vision sensor platforms based on CMOS technology act as transduction entities that convert incident light intensities into digitized pixel values, through a two-dimensional array of photodiodes [3]. The vision data generated from such CMOS Image Sensors (CIS) are often processed elsewhere in a cloud environment consisting of CPUs and GPUs [11]. This physical segregation leads to bottlenecks in throughput, bandwidth, and energy-efficiency for applications that require transferring large amounts of data from the image sensor to the back-end processor, such as object detection and tracking from high-resolution images/videos.

To address these bottlenecks, many researchers are trying to bring intelligent data processing closer to the source of the vision data, *i.e.,* closer to the CIS, taking one of

three broad approaches - near-sensor processing [9, 10], in-sensor processing [12], and in-pixel processing [13–15]. Near-sensor processing aims to incorporate a dedicated machine learning accelerator chip on the same printed circuit board [9], or even 3D-stacked with the CIS chip [10]. Although this enables processing of the CIS data closer to the sensor rather than in the cloud, it still suffers from the data transfer costs between the CIS and processing chip. On the other hand, in-sensor processing solutions [12] integrate digital or analog circuits within the periphery of the CIS sensor chip, reducing the data transfer between the CIS sensor and processing chips. Nevertheless, these approaches still often require data to be streamed (or read in parallel) through a bus from CIS photo-diode arrays into the peripheral processing circuits [12]. In contrast, in-pixel processing solutions, such as [13–15, 37, 38], aim to embed processing capabilities within the individual CIS pixels. Initial efforts have focused on in-pixel analog convolution operation [37, 38] but many [13, 37–39] require the use of emerging non-volatile memories or 2D materials. Unfortunately, these technologies are not yet mature and thus not amenable to the existing foundry-manufacturing of CIS. Moreover, these works fail to support multi-bit, multi-channel convolution operations, batch normalization (BN), and Rectified Linear Units (ReLU) needed for most practical deep learning applications. Furthermore, works that target digital CMOS-based in-pixel hardware, organized as pixel-parallel single instruction multiple data (SIMD) processor arrays [14], do not support convolution operation, and are thus limited to toy workloads, such as digit recognition. Many of these works rely on digital processing which typically yields lower levels of parallelism compared to their analog in-pixel alternatives. In contrast, the work in [15], leverages in-pixel parallel analog computing, wherein the weights of a neural network are represented as the exposure time of individual pixels. Their approach requires weights to be made available for manipulating pixel-exposure time through control pulses, leading to a data transfer bottleneck between the weight memories and the sensor array. Thus, an in-situ CIS processing solution where both the weights and input activations are available within individual pixels that efficiently implements critical deep learning operations such as multi-bit, multi-channel convolution, BN, and ReLU operations has remained elusive. Furthermore, all existing in-pixel computing solutions have targeted datasets that do not represent realistic applications of machine intelligence mapped onto state-of-the-art CIS. Specifically, most of the existing works are focused on simplistic datasets like MNIST [14], while few [15] use the CIFAR-10 dataset which has input images with a significantly low resolution (32×32), that does not represent images captured by state-of-

the-art high resolution CIS.

Towards that end, we propose a novel in-situ computing paradigm at the sensor nodes called *Processing-in-Pixel-in-Memory* (P^2M), illustrated in Fig. 3.1, that incorporates both the network weights and activations to enable massively parallel, high-throughput intelligent computing inside CISs. In particular, our circuit architecture not only enables in-situ multi-bit, multi-channel, dot product analog acceleration needed for convolution, but re-purposes the on-chip digital *correlated double sampling* (CDS) circuit and single slope ADC (SS-ADC) typically available in conventional CIS to implement *all the required computational aspects for the first few layers* of a state-of-the-art deep learning network. Furthermore, the proposed architecture is coupled with a circuit-algorithm co-design paradigm that captures the circuit non-linearities, limitations, and bandwidth reduction goals for improved latency and energy-efficiency. The resulting paradigm is the first to demonstrate the feasibility for enabling complex, intelligent image processing applications (beyond toy datasets), on high-resolution images of Visual Wake Words (VWW) dataset, catering to a real-life TinyML application. We choose to evaluate the efficacy of P^2M on TinyML applications, as they impose tight compute and memory budgets, that are otherwise difficult to meet with current in- and near-sensor processing solutions, particularly for high-resolution input images. Key highlights of the presented work are as follows:

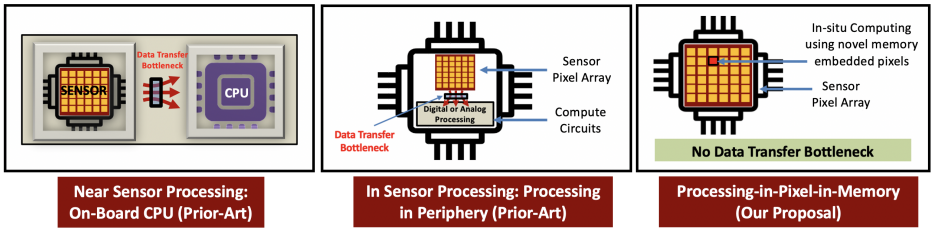


Figure 3.1: Existing and Proposed Solutions to alleviate the energy, throughput, and bandwidth bottleneck caused by the segregation of *Sensing* and *Compute*.

3.2 Challenges & Opportunities in P^2M

The ubiquitous presence of CIS-based vision sensors has driven the need to enable machine learning computations closer to the sensor nodes. However, given the computing complexity of modern CNNs, such as Resnet-18 [20] and SqueezeNet [40], it is not feasible to execute the entire deep-learning network, including all the layers within the CIS chip. As a result,

recent intelligent vision sensors, for example, from Sony [10], which is equipped with basic AI processing functionality (e.g., computing image metadata), features a multi-stacked configuration consisting of separate pixel and logic chips that must rely on high and relatively energy-expensive inter-chip communication bandwidth.

Alternatively, we assert that embedding part of the deep learning network within pixel arrays in an in-situ manner can lead to a significant reduction in data bandwidth (and hence energy consumption) between sensor chip and downstream processing for the rest of the convolutional layers. This is because the first few layers of carefully designed CNNs, as explained in Section 3.4, can have a significant compressing property, i.e., the output feature maps have reduced bandwidth/dimensionality compared to the input image frames. In particular, our proposed P^2M paradigm enables us to map all the computations of the first few layers of a CNN into the pixel array. The paradigm includes a holistic hardware-algorithm co-design framework that captures the specific circuit behavior, including circuit non-idealities, and hardware limitations, during the design, optimization, and training of the proposed machine learning networks. The trained weights for the first few network layers are then mapped to specific transistor sizes in the pixel-array. Because the transistor widths are fixed during manufacturing, the corresponding CNN weights lack programmability. Fortunately, it is common to use the pre-trained versions of the first few layers of modern CNNs as high-level feature extractors are common across many vision tasks [18].

3.3 P^2M Circuit Implementation

This section describes key circuit innovations that enable us to embed all the computational aspects for the first few layers of a complex CNN architecture within the CIS. An overview of our proposed pixel array that enables the availability of weights and activations within individual pixels with appropriate peripheral circuits is shown in Fig. 3.2.

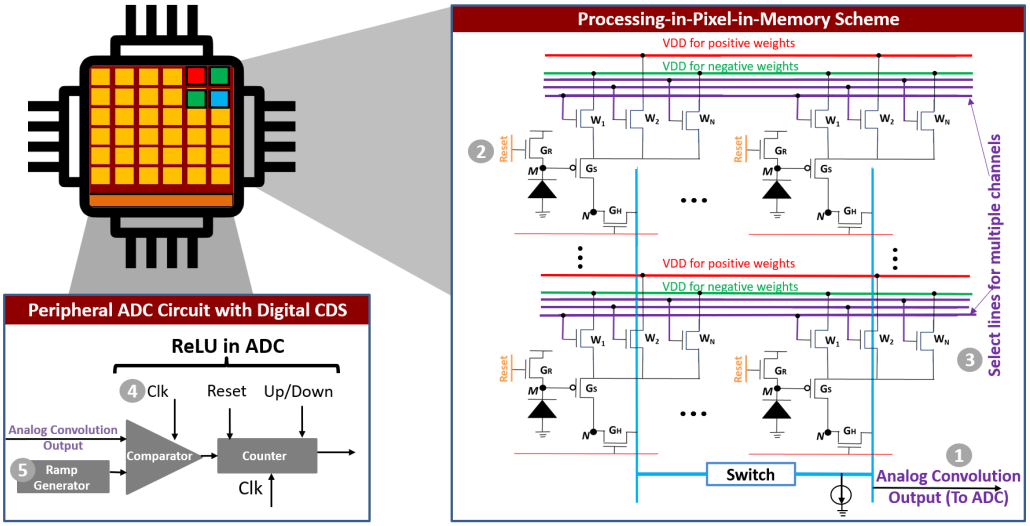
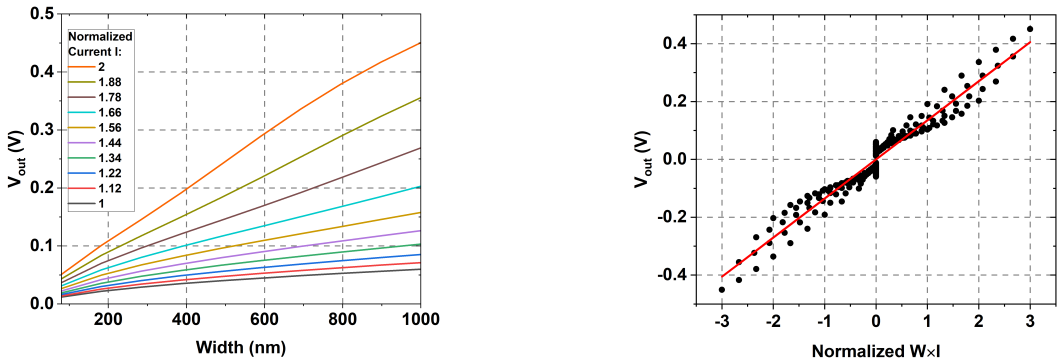


Figure 3.2: Proposed circuit techniques based on presented P^{2}M scheme capable of mapping all computational aspects for the first few layers of a modern CNN layer within CIS pixel arrays.

3.3.1 Multi-Channel, Multi-Bit Weight Embedded Pixels

Our modified pixel circuit builds upon the standard three transistor pixel by embedding additional transistors W_is that represent weights of the CNN layer, as shown in Fig. 3.2. Each weight transistor W_i is connected in series with the source-follower transistor G_s. When a particular weight transistor W_i is activated (by pulling its gate voltage to V_{DD}), the pixel output is modulated both by the driving strength of the transistor W_i and the voltage at the gate of the source-follower transistor G_s. A higher photo-diode current implies the PMOS source follower is strongly ON, resulting in an increase in the output pixel voltage. Similarly, a higher width of the weight transistor W_i results in lower transistor resistance and hence lower source degeneration for the source follower transistor, resulting in higher pixel output voltage. Fig. 3.3(a), obtained from SPICE simulations using 22nm GlobalFoundries technology exhibits the desired dependence on transistor width and input photo-diode current. Thus, the pixel output performs an approximate multiplication of the input light intensity (voltage at the gate of transistor G_s) and the weight (or driving strength) of the transistor W_i, as exhibited by the plot in Fig. 3.3(b). The approximation stems from the fact that transistors are inherently non-linear. In Section 4, we leverage our hardware-algorithm co-design framework to incorporate the circuit non-linearities within

the CNN training framework, thereby maintaining close to state-of-the-art classification accuracy. Multiple weight transistors W_is are incorporated within the same pixel and are controlled by independent gate control signals. Each weight transistor implements a different channel in the output feature map of the layer. Thus, the gate signals represent select lines for specific channels in the output feature map. Note, it is desirable to reduce the number of output channels so as to reduce the total number of weight transistors embedded within each pixel while ensuring high test accuracy for VWW. For our work, using a holistic hardware-algorithm co-design framework (Section 3.5.2), we were able to reduce the number of channels in the first layer from 16 to 8, this implies the proposed circuit requires 8 weight transistors per pixel, which can be reasonably implemented.



(a) **(b)**

Figure 3.3: (a) Pixel output voltage as a function of weight (transistor width) and input activation (Normalized photo-diode current) simulated on Globalfoundries 22nm FD-SOI node. As expected pixel output increases both as a function of weights and input activation. (b) A scatter plot comparing pixel output voltage to ideal multiplication value of Weights×Input activation (Normalized $W \times I$). The plot confirms that the output pixel voltage from each pixel represents approximate product of Weights×Input activation.

3.3.2 In-situ Multi-pixel Convolution Operation

To achieve the convolution operation, we simultaneously activate multiple pixels. In the specific case of VWW, we activate $X \times Y \times 3$ pixels at the same time, where X and Y denote the spatial dimensions and 3 corresponds to the RGB (red, blue, green) channels in the input activation layer. For each activated pixels, the pixel output is modulated by the photo-diode current and the weight of the activated W_i transistor associated with the pixel, in

accordance with Fig. 3.3(a)-(b). For a given convolution operation only one weight transistor is activated per pixel, corresponding to a specific channel in the first layer of the CNN. The weight transistors W_i represent multi-bit weights through their driving strength. As detailed in Section 3.1, for each pixel, the output voltage approximates the multiplication of light intensity and weight. For each bit line, shown as vertical blue lines in Fig. 3.2, the cumulative pull up strength of the activated pixels connected to that line drives it high. The increase in pixel output voltages accumulate on the bit lines implementing an analog summation operation. Consequently, the voltage at the output of the bit lines represent the convolution operation between input activations and the stored weight inside the pixel.

Fig. 3.4 plots the output voltage (at node *Analog Convolution Output* in Fig. 3.2) as a function of normalized ideal convolution operation. The plot in the figure was generated by considering 75 pixels are activated, simultaneously. For each line in Fig. 3.4, the activated weight transistors W_i are chosen to have the same width and the set of colored lines represents the range of widths. For each line, the input I is swept from its minimum to maximum value and the ideal dot product is normalized and plotted on x-axis. The y-axis plots the actual SPICE circuit output. The largely linear nature of the plot indicates that the circuits are working as expected and the small amount of non-linearities are captured in our training framework described in Section 4.1.

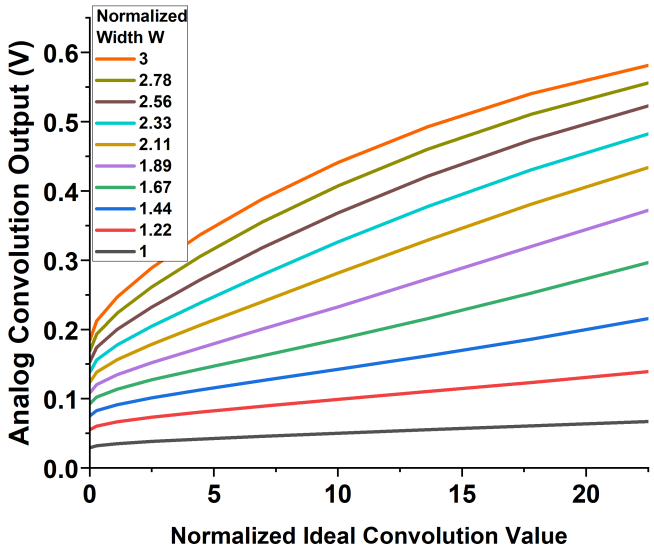
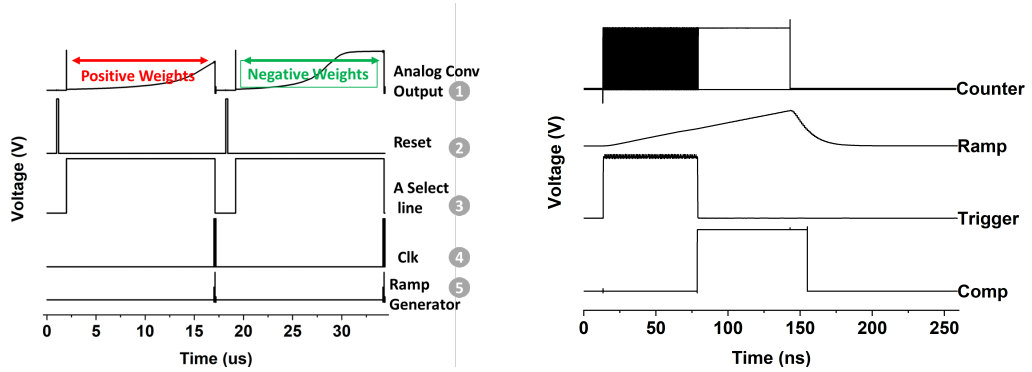


Figure 3.4: Output voltage at the *Analog Convolution Output* node as a function of the normalized ideal convolution operation with 75 pixels activated simultaneously. Each line corresponds to a specific transistor width (weight), and the input I is swept from minimum to maximum. The largely linear relationship confirms the expected circuit behavior; residual non-linearities are captured by the training framework.

3.3.3 Re-purposing Digital Correlated Double Sampling Circuit and Single-Slope ADCs as ReLU Neurons

Weights in a CNN layer span positive and negative values. As discussed in the previous sub-section, weights are mapped by the driving strength (or width) of transistors W_is. As the width of transistors cannot be negative, the W_i transistors themselves cannot represent negative weights. Interestingly, we circumvent this issue by re-purposing on-chip digital CDS circuit present in many state-of-the-art commercial CIS [7, 8]. A digital CDS is usually implemented in conjunction to column parallel Single Slope ADCs (SS-ADCs). A single slope ADC consists of a ramp-generator, a comparator, and a counter (see Fig. 3.2). An input analog voltage is compared through the comparator to a ramping voltage with a fixed slope, generated by the ramp generator. A counter which is initially reset, and supplied with an appropriate clock, keeps counting until the ramp voltage crosses the analog input voltage. At this point, the output of counter is latched and represents the converted digital

value for input analog voltage. A traditional CIS digital CDS circuit takes as input two correlated samples at two different time instances. The first sample corresponds to the reset noise of the pixel and the second sample to the actual signal superimposed with the reset noise. A digital CIS CDS circuit then takes the difference between the two samples, thereby, eliminating reset noise during ADC conversion. In an SS-ADC the difference is taken by simply making the counter 'up' count for one sample and 'down' count for the second.



(a) **(b)**

Figure 3.5: (a) A typical timing waveform, showing double sampling (one for positive and other for negative) weights. The numerical labels in the figure correspond to the numerical label in the circuit shown in Fig. 3.2. (b) Typical timing waveform for the SS-ADC showing comparator output (Comp), counter enable (trigger), ramp generator output, and counter clock (Counter).

We utilize the noise cancelling, differencing behavior of the CIS digital CDS circuit already available on commercial CIS chips to implement positive and negative weights and implement ReLU. First, each weight transistor embedded inside a pixel is 'tagged' as a positive or a 'negative weight' by connecting it to 'red lines' (marked as VDD for positive weights in Fig. 3.2) and 'green lines' (marked as VDD for negative weights in Fig. 3.2). For each channel, we activate multiple pixels to perform an inner-product and read out two samples. The first sample corresponds to a high VDD voltage applied on the 'red lines' (marked as VDD for positive weights in Fig. 3.2) while the 'green lines' (marked as VDD for negative weights in Fig. 3.2) are kept at ground. The accumulated multi-bit dot product result is digitized by the SS-ADC, while the counter is 'up' counting. The second sample, on the other hand, corresponds to a high VDD voltage applied on the 'green lines' (marked as VDD for negative weights in Fig. 3.2) while the 'red lines' (marked as VDD for positive

weights in Fig. 3.2) are kept at ground. The accumulated multi-bit dot product result is again digitized and also subtracted from the first sample by the SS-ADC, while the counter is 'down' counting. Thus, the digital CDS circuit first accumulates the convolution output for all positive weights and then subtracts the convolution output for all negative weights for each channel, controlled by respective select lines for individual channels. Note, possible sneak currents flowing between weight transistors representing positive and negative weights can be obviated by integrating a diode in series with weight transistors or by simply splitting each weight transistor into two series connected transistors, where the channel select lines control one of the series connected transistor, while the other transistor is controlled by a select line representing positive/negative weights.

Interestingly, re-purposing the on-chip CDS for implementing positive and negative weights also allows us to easily implement a quantized ReLU operation inside the SS-ADC. ReLU clips negative values to zero. This can be achieved by ensuring that the final count value latched from the counter (after the CDS operation consisting of 'up' counting and then 'down' counting') is either positive or zero. Interestingly, before performing the dot product operation, the counter can be reset to a non-zero value representing the scale factor of the BN layer as described in Section 3.4. Thus, by embedding multi-pixel convolution operation and re-purposing on-chip CDS and SS-ADC circuit for implementing positive/negative weights, batch-normalization and ReLU operation, our proposed P^2M scheme can implement all the computational aspect for the first few layers of a complex CNN within the pixel array enabling massively parallel in-situ computations.

Putting these features together, our proposed P^2M circuit computes one channel at a time and has three phases of operation:

1. Reset Phase: First, the voltage on the photodiode node M (see Fig. 3.2) is pre-charged or reset by activating the reset transistor G_r. Note, since we aim at performing multi-pixel convolution, the set of pixels $X \times Y \times 3$ are reset, simultaneosuly.

2. Multi-pixel Convolution Phase: Next, we discharge the gate of the reset transistor G_r which deactivates G_r. Subsequently, $X \times Y \times 3$ pixels are activated by pulling the gate of respective G_H transistors to VDD. Within the activated set of pixels, a single weight transistor corresponding to a particular channel in the output feature map is activated, by pulling high its gate voltage through the select lines (labeled as select lines for multiple channels in Fig. 2). As the photodiode is sensitive to the incident light, photo-

current is generated as light shines upon the diode (for a duration equal to exposure time), and voltage on the gate of G_s is modulated in accordance to the photodiode current that is proportional to the intensity of incident light. The pixel output voltage is a function of the incident light (voltage on node M) and the driving strength of the activated weight transistor within each pixel. Pixel output from multiple pixels are accumulated on the column-lines and represent the multi-pixel analog convolution output. The SS-ADC in the periphery converts analog output to a digital value. Note, the entire operation is repeated twice, one for positive weights ('up' counting) and another for negative weights ('down counting').

3. ReLU Operation: Finally, the output of the counter is latched and represents a quantized ReLU output. It is ensured that the latched output is either positive or zero, thereby mimicking the ReLU functionality within the SS-ADC.

The entire P^2M circuit is simulated using commercial 22nm GlobalFoundries FD-SOI (fully depleted silicon-on-insulator) technology, the SS-ADCs are implemented using a using a bootstrap ramp generator and dynamic comparators. Assuming the counter output which represents the ReLU function is an N-bit integer, it needs 2^N cycles for a single conversion. The ADC is supplied with a 2GHz clock for the counter circuit. SPICE simulations exhibiting the multiplicative nature of weight transistor embedded pixels with respect to photodiode current is shown in Fig. 3.3(a)-3.3(b). Functional behavior of the circuit for analog convolution operation is depicted in Fig. 3.4. A typical timing waveform showing pixel operation along with SS-ADC operation simulated on 22nm GlobalFoundries technology node is shown in Fig. 3.5.

It may also be important to note that a highlight of our proposal is that we re-purpose various circuit functions already available in commercial cameras. This ensures most of the existing peripheral and corresponding timing control blocks would require only minor modification to support our proposed P^2M computations. Specifically, instead of activating one row at a time in a rolling shutter manner, P^2M requires activation of group of rows, simultaneously, corresponding to the size of kernels in the first layers. Multiple group of rows would then be activated in a typical rolling shutter format. Overall, the sequencing of pixel activation (except for the fact that group of rows have to be activated instead of a single row), CDS, ADC operation and bus-readout would be similar to typical cameras [4].

3.3.4 CIS Process Integration and Area Considerations

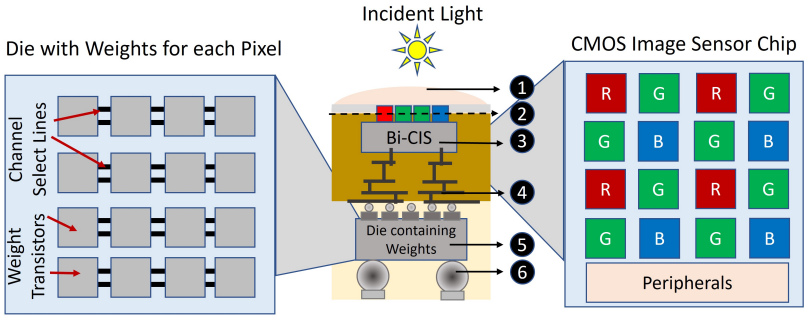


Figure 3.6: representative illustration of heterogeneously integrated system featuring P^{2}M paradigm, built on backside illuminated CMOS image sensor (Bi-CIS). ① Micro lens, ② Light shield, ③ Backside illuminated CMOS Image Sensor (Bi-CIS), ④ Backend of line of the Bi-CIS, ⑤ Die consisting of weight transistors, ⑥ solder bumps for input/output bus (I/O).

In this section, we would like to highlight the viability of the proposed P^{2}M paradigm featuring memory-embedded pixels with respect to its manufacturability using existing foundry processes. A representative illustration of a heterogeneously integrated system catering to the needs of the proposed P^{2}M paradigm is shown in Fig. 3.6. The figure consists of two key elements, i) backside illuminated CMOS image sensor (Bi-CIS), consisting of photo-diodes, read-out circuits and pixel transistors (reset, source follower and select transistors), and ii) a die consisting of multiple weight transistors per pixel (refer Fig 3.2). From Fig. 3.2, it can be seen that each pixel consists of multiple weight transistors that would lead to exceptionally high area overhead. However, with the presented heterogeneous integration scheme of Fig. 3.6, the weight transistors are vertically aligned below a standard pixel, thereby incurring no (or minimal) increase in footprint. Specifically, each Bi-CIS chip can be implemented in a leading or lagging technology node. The die consisting of weight transistors can be built on an advanced planar or non-planar technology node such that the multiple weight transistors can be accommodated in the same footprint occupied by a single pixel (assuming pixel sizes are larger than the weight transistor embedded memory circuit configuration). The Bi-CIS image sensor chip/die is heterogeneously integrated through a bonding process (die-to-die or die-to-wafer) integrating it onto the die consisting of weight transistors. Preferably, a die-to-wafer low-temperature metal-to-metal fusion with a dielectric-to-dielectric direct bonding hybrid process can achieve high-throughput

sub-micron pitch scaling with precise vertical alignment [41] . One of the advantages of adapting this heterogeneous integration technology is that chips of different sizes can be fabricated at distinct foundry sources, technology nodes, and functions and then integrated together. In case there are any limitations due to the increased number of transistors in the die consisting of the weights, a conventional pixel-level integration scheme, such as Stacked Pixel Level Connections (SPLC), which shields the logic CMOS layer from the incident light through the Bi-CIS chip region, would also provide a high pixel density and a large dynamic range [42]. Alternatively, one could also adopt the *through silicon via* (TSV) integration technique for front-side illuminated CMOS image sensor (Fi-CIS), wherein the CMOS image sensor is bonded onto the die consisting of memory elements through a TSV process. However, in the Bi-CIS, the wiring is moved away from the illuminated light path allowing more light to reach the sensor, giving better low-light performance [43].

Advantageously, the heterogeneous integration scheme can be used to manufacture P^2M sensor systems on existing as well as emerging technologies. Specifically, the die consisting of weight transistors could use a ROM-based structure as shown in Section 3.3 or other emerging programmable non-volatile memory technologies like PCM [44], RRAM [45], MRAM [46], ferroelectric field effect transistors (FeFETs) [47] etc., manufactured in distinct foundries and subsequently heterogeneously integrated with the CIS die. Thus, the proposed heterogeneous integration allows us to achieve lower area-overhead, while simultaneously enabling seamless, massively parallel convolution. Specifically, based on reported contacted poly pitch and metal pitch numbers [48], we estimate more than 100 weight transistors can be embedded in a 3D integrated die using a 22nm technology, assuming the underlying pixel area (dominated by the photodiode) is $10\mu m \times 10\mu m$. Availability of back-end-of-line monolithically integrated two terminal non-volatile memory devices could allow denser integration of weights within each pixel. Such weight embedded pixels allow individual pixels to have in-situ access to both activation and weights as needed by the P^2M paradigm which obviates the need to transfer weights or activation from one physical location to another through a bandwidth constrained bus. Hence, unlike other multi-chip solutions [10], our approach does not incur energy bottlenecks.

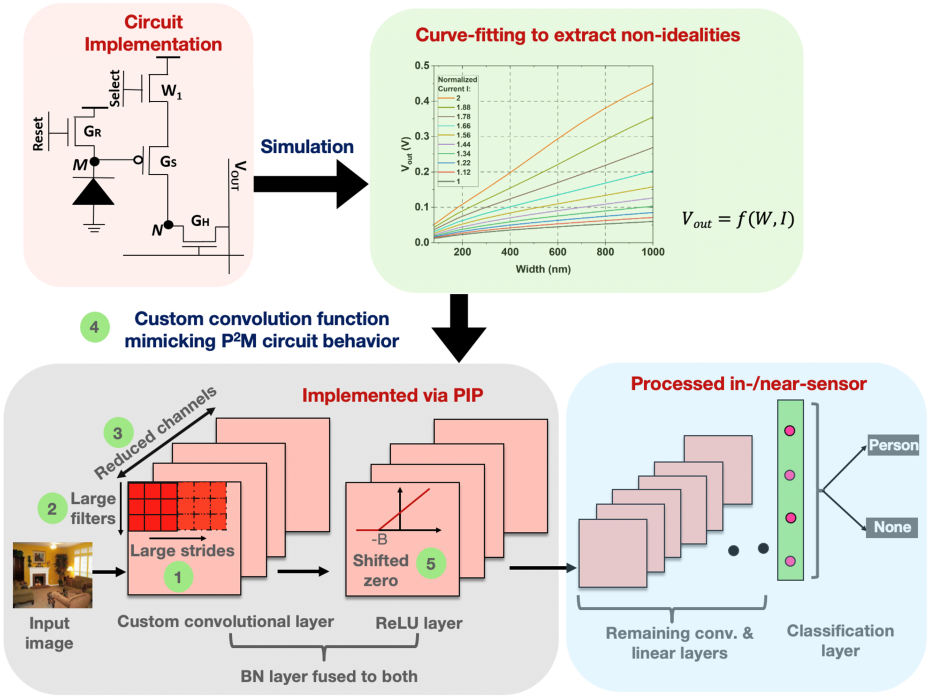


Figure 3.7: Algorithm-circuit co-design framework to enable our proposed P^{2}M approach optimize both the performance and energy-efficiency of vision workloads. We propose the use of ① large strides, ② large kernel sizes, ③reduced number of channels, ④ P^{2}M custom convolution, and ⑤ shifted ReLU operation to incorporate the shift term of the batch normalization layer, for emulating accurate P^{2}M circuit behaviour.

3.4 P^{2}M-constrained Algorithm-Circuit Co-Design

In this section, we present our algorithmic optimizations to standard CNN backbones that are guided by 1) P^{2}M circuit constraints arising due to analog computing nature of the proposed pixel array and the limited conversion precision of on-chip SS-ADCs, 2) the need for achieving state-of-the-art test accuracy, and 3) maximizing desired hardware metrics of high bandwidth reduction, energy-efficiency and low-latency of P^{2}M computing, and meeting the memory and compute budget of the VWW application. The reported improvement in hardware metrics (illustrated in Section 3.5.3), is thus a result of intricate circuit-algorithm co-optimization.

3.4.1 Custom Convolution for the First Layer Modeling Circuit Non-Idealities

From an algorithmic perspective, the first layer of a CNN is a linear convolution operation followed by BN, and non-linear (ReLU) activation. The P^2M circuit scheme, explained in Section 3.3, implements convolution operation in analog domain using modified memory-embedded pixels. The constituent entities of these pixels are transistors, which are inherently non-linear devices. As such, in general, any analog convolution circuit consisting of transistor devices will exhibit non-ideal non-linear behavior with respect to the convolution operation. Many existing works, specifically in the domain of memristive analog dot product operation, ignore non-idealities arising from non-linear transistor devices [26, 27]. In contrast, to capture these non-linearities, we performed extensive simulations of the presented P^2M circuit spanning wide range of circuit parameters such as the width of weight transistors and the photodiode current based on commercial 22nm Globafoundries transistor technology node. The resulting SPICE results, *i.e.* the pixel output voltages corresponding to a range of weights and photodiode currents, were modeled using a behavioral curve-fitting function. The generated function was then included in our algorithmic framework, replacing the convolution operation in the first layer of the network. In particular, we accumulate the output of the curve-fitting function, one for each pixel in the receptive field (we have 3 input channels, and a kernel size of 5×5, and hence, our receptive field size is 75), to model each inner-product generated by the in-pixel convolutional layer. This algorithmic framework was then used to optimize the CNN training for the VWW dataset.

3.4.2 Circuit-Algorithm Co-optimization of CNN Backbone subject to P^2M Constrains

As explained in Section 3.3.1, the P^2M circuit scheme maximizes parallelism and data bandwidth reduction by activating multiple pixels and reading multiple parallel analog convolution operations for a given channel in the output feature map. The analog convolution operation is repeated for each channel in the output feature map serially. Thus, parallel convolution in the circuit tends to improve parallelism, bandwidth reduction, energy-efficiency and speed. But, increasing the number of channels in the first layer increases the serial aspect of the convolution and degrades parallelism, bandwidth reduction, energy-efficiency,

and speed. This creates an intricate circuit-algorithm trade-off, wherein the backbone CNN has to be optimized for having larger kernel sizes (that increases the concurrent activation of more pixels, helping parallelism) and non-overlapping strides (to reduce the dimensionality in the downstream CNN layers, thereby reducing the number of multiply-and-adds and peak memory usage), smaller number of channels (to reduce serial operation for each channel), while maintaining close to state-of-the-art classification accuracy and taking into account the non-idealities associated with analog convolution operation. Also, decreasing number of channels decreases the number of weight transistors embedded within each pixel (each pixel has weight transistors equal to the number of channels in the output feature map), improving area and power consumption. Furthermore, the resulting smaller output activation map (due to reduced number of channels, and larger kernel sizes with non-overlapping strides) reduces the energy incurred in transmission of data from the CIS to the downstream CNN processing unit and the number of floating point operations (and consequently, energy consumption) in downstream layers.

In addition, we propose to fuse the BN layer, partly in the preceding convolutional layer, and partly in the succeeding ReLU layer to enable its implementation via P^2M. Let us consider a BN layer with γ and β as the trainable parameters, which remain fixed during inference. During the training phase, the BN layer normalizes feature maps with a running mean μ and a running variance σ, which are saved and used for inference. As a result, the BN layer implements a linear function, as shown below.

$$Y = \gamma \frac{X-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta = \left(\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}\right) \cdot X + \left(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}\right) = A \cdot X + B \tag{3.1}$$

We propose to fuse the scale term A into the weights (value of the pixel embedded weight tensor is $A \cdot \theta$, where θ is the final weight tensor obtained by our training) that are embedded as the transistor widths in the pixel array. Additionally, we propose to use a shifted ReLU activation function, following the covolutional layer, as shown in Fig. 3.7 to incorporate the shift term B. We use the counter-based ADC implementation illustrated in Section 3.3.3 to implement the shifted ReLU activation. This can be easily achieved by resetting the counter to a non-zero value corresponding to the term B at the start of the convolution operation, as opposed to resetting the counter to zero.

Moreover, to minimize the energy cost of the analog-to-digital conversion in our P^2M approach, we must also quantize the layer output to as few bits as possible subject to

achieving the desired accuracy. We train a floating-point model with close to state-of-the-accuracy, and then perform quantization in the first convolutional layer to obtain low-precision weights and activations during inference [49]. We also quantize the mean, variance, and the trainable parameters of the BN layer, as all these affect the shift term B (please see Eq. 3.1), that should be quantized for the low-precision shifted ADC implementation. We avoid quantization-aware training [50] because it significantly increases the training cost with no reduction in bit-precision for our model at iso-accuracy. Note that the lack of bit-precision improvement from QAT is probably because a small improvement in quantization of only the first layer may have little impact on the test accuracy of the whole network.

With the bandwidth reduction obtained by all these approaches, the output feature map of the P^2M-implemented layers can more easily be implemented in micro-controllers with extremely low memory footprint, while P^2M itself greatly improves the energy-efficiency of the first layer. Our approach can thus enable TinyML applications that usually have a tight compute and memory budget, as illustrated in Section 3.5.1

3.4.3 Quantification of bandwidth reduction

To quantify the bandwidth reduction (BR) after the first layer obtained by P^2M (BN and ReLU layers do not yield any BR), let the number of elements in the RGB input image be I and in the output activation map after the ReLU activation layer be O. Then, BR can be estimated as

$$BR = \left(\frac{O}{I}\right)\left(\frac{4}{3}\right)\left(\frac{12}{N_b}\right) \tag{3.2}$$

Here, the factor $\left(\frac{4}{3}\right)$ represents the compression from Bayer's pattern of RGGB pixels to RGB pixels because we can either ignore the additional green pixel or design the circuit to effectively take the average of the photo-diode currents from the two green pixels. The factor $\frac{12}{N_b}$ represents the ratio of the bit-precision between the image pixels captured by the sensor (pixels typically have a bit-depth of 12 [25]) and the quantized output of our convolutional layer denoted as N_b. Let us now substitute

$$O = \left(\frac{i-k+2*p}{s}+1\right)^2 * c_o, \quad I = i^2 * 3 \tag{3.3}$$

into Eq. 3.2, where i denotes the spatial dimension of the input image, k, p, s denote the kernel size, padding and stride of the in-pixel convolutional layer, respectively, and c_o denotes the number of output channels of the in-pixel convolutional layer. These hyperparameters, along with N_b are obtained via a thorough algorithmic design space exploration with the goal of achieving the best accuracy, subject to meeting the hardware constraints and the memory and compute budget of our TinyML benchmark. We show their values in Table 3.1, and substitute them in Eq. 3.2 to obtain a BR of 21×.

Hyperparameter	Value
kernel size of the convolutional layer (k)	5
padding of the convolutional layer (p)	0
stride of the convolutional layer (s)	5
number of output channels of the convolutional layer (c_o)	8
bit-precision of the P^{2}M-enabled convolutional layer output (N_b)	8

Table 3.1: Model hyperparameters and their values to enable bandwidth reduction in the in-pixel layer.

3.5 Experimental Results

3.5.1 Benchmarking Dataset & Model

This paper focuses on the potential of P^{2}M for TinyML applications, *i.e.*, with models that can be deployed on low-power IoT devices with only a few kilobytes of on-chip memory [51–53]. In particular, the Visual Wake Words (VWW) dataset [54] presents a relevant use case for visual TinyML. It consists of high resolution images that include visual cues to "wake-up" AI-powered home assistant devices, such as Amazon's Astro [55], that requires real-time inference in resource-constrained settings. The goal of the VWW challenge is to detect the presence of a human in the frame with very little resources - close to 250KB peak RAM usage and model size [54]. To meet these constraints, current solutions involve downsampling the input image to medium resolution (224×224) which costs some accuracy [49].

In this work, we use the images from the COCO2014 dataset [56] and the train-val split specified in the seminal paper [54] that introduced the VWW dataset. This split ensures that

the training and validation labels are roughly balanced between the two classes 'person' and 'background'; 47% of the images in the training dataset of 115k images have the 'person' label, and similarly, 47% of the images in the validation dataset are labelled to the 'person' category. The authors also ensure that the distribution of the area of the bounding boxes of the 'person' label remain similar across the train and val set. Hence, the VWW dataset with such a train-val split acts as the primary benchmark of tinyML models [57] running on low-power microcontrollers. We choose MobileNetV2 [21] as our baseline CNN architecture with 32 and 320 channels for the first and last convolutional layers respectively that supports full resolution (560×560) images. In order to avoid overfitting to only two classes in the VWW dataset, we decrease the number of channels in the last depthwise separable convolutional block by 3×. MobileNetV2, similar to other MobileNet class of models, is very compact [21] with size less than the maximum allowed in the VWW challenge. It performs well on complex datasets like ImageNet [58] and, as shown in Section 3.5, does very well on VWWs.

To evaluate P^2M on MobileNetV2, we create a custom model that replaces the first convolutional layer with our P^2M custom layer that captures the systematic non-idealities of the analog circuits, the reduced number of output channels, and limitation of non-overlapping strides, as discussed in Section 3.4.

We train both the baseline and P^2M custom models in PyTorch using the SGD optimizer with momentum equal to 0.9 for 100 epochs. The baseline model has an initial learning rate (LR) of 0.03, while the custom counterpart has an initial LR of 0.003. Both the learning rates decay by a factor of 0.2 at every 35 and 45 epochs. After training a floating-point model with the best validation accuracy, we perform quantization to obtain 8-bit integer weights, activations, and the parameters (including the mean and variance) of the BN layer. All experiments are performed on a Nvidia 2080Ti GPU with 11 GB memory.

3.5.2 Classification Accuracy

Comparison between baseline and P^2M custom models: We evaluated the performance of the baseline and P^2M custom MobileNet-V2 models on the VWW dataset in Table 3.2. Note that both these models are trained from scratch. Our baseline model currently yields the best test accuracy on the VWW dataset among the models available in literature that does not leverage any additional pre-training or augmentation. Note that our baseline model requires a significant amount of peak memory and MAdds (~30× more than that allowed

in the VWW challenge), however, serves a good benchmark for comparing accuracy. We observe that the P^{2}M-enabled custom model can reduce the number of MAdds by ∼7.15×, and peak memory usage by ∼25.1× with 1.47% drop in the test accuracy compared to the uncompressed baseline model for an image resolution of 560×560. With the memory reduction, our P^{2}M model can run on tiny micro-controllers with only 270 KB of on-chip SRAM. Note that peak memory usage is calculated using the same convention as [54]. Notice also that both the baseline and custom model accuracies drop (albeit the drop is significantly higher for the custom model) as we reduce the image resolution, which highlights the need for high-resolution images and the efficacy of P^{2}M in both alleviating the bandwidth bottleneck between sensing and processing, and reducing the number of MAdds for the downstream CNN processing.

Image Resolution	Model	Test Accuracy (%)	Number of MAdds (G)	Peak memory usage (MB)
560× 560	baseline	91.37	1.93	7.53
	in-sensor	89.90	0.27	0.30
225× 225	baseline	90.56	0.31	1.2
	in-sensor	84.30	0.05	0.049
115× 115	baseline	91.10	0.09	0.311
	in-sensor	80.00	0.01	0.013

Table 3.2: Test accuracies, number of MAdds, and peak memory usage of baseline and P^{2}M custom compressed model while classifying on the VWW dataset for different input image resolutions.

Comparison with SOTA models: Table 3.3 provides a comparison of the performances of models generated through our algorithm-circuit co-simulation framework with SOTA TinyML models for VWW. Our P^{2}M custom models yield test accuracies within 0.37% of the best performing model in the literature [59]. Note that we have trained our models solely based on the training data provided, whereas ProxylessNAS [59], that won the 2019 VWW challenge leveraged additional pretraining with ImageNet. Hence, for consistency, we report the test accuracy of ProxylessNAS with identical training configurations on the final network provided by the authors, similar to [49]. Note that [60] leveraged massively parallel energy-efficient analog in-memory computing to implement MobileNet-V2 for VWW, but incurs an accuracy drop of 5.67% and 4.43% compared to our baseline and the previous state-of-the-art [59] models. This probably implies the need for intricate algorithm-hardware co-design

and accurately modeling of the hardware non-idealities in the algorithmic framework, as shown in our work.

Authors	Description	Model Architecture	Test Accuracy (%)
Saha et al. (2020) [49]	RNNPooling	MobileNetV2	89.65
Han et al. (2019) [59]	ProxylessNAS	Non-standard architecture	90.27
Banbury et al. (2021) [53]	Differentiable NAS	MobileNet-V2	88.75
Zhou et al. (2021) [60]	Analog compute-in-memory	MobileNet-V2	85.7
This work	P^{2}M	MobileNet-V2	89.90

Table 3.3: Performance comparison of the proposed P^{2}M-compatible models with state-of-the-art deep CNNs on VWW dataset.

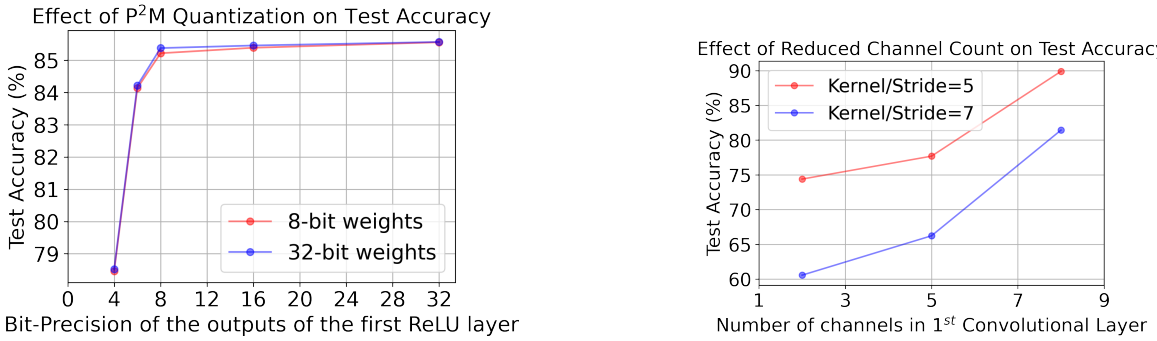


Figure 3.8: (a) Effect of quantization of the in-pixel output activations, and (b) Effect of the number of channels in the 1st convolutional layer for different kernel sizes and strides, on the test accuracy of our P^{2}M custom model.

Effect of quantization of the in-pixel layer: As discussed in Section 3.4, we quantize the output of the first convolutional layer of our proposed model after training to reduce the power consumption due to the sensor ADCs and compress the output as outlined in Eq. 3.2. We sweep across output bit-precisions of {4,6,8,16,32} to explore the trade-off between accuracy and compression/efficiency as shown in Fig. 3.8(a). We choose a bit-width of 8 as it is the lowest precision that does not yield any accuracy drop compared to the full-precision models. As shown in Fig. 3.8, the weights in the in-pixel layer can also be quantized to 8 bits with an 8-bit output activation map, with less than 0.1% drop in accuracy.

Ablation study: We also study the accuracy drop incurred due to each of the three modifications (non-overlapping strides, reduced channels, and custom function) in the P^{2}M-

enabled custom model. Incorporation of the non-overlapping strides (stride of 5 for 5×5 kernels from a stride of 2 for 3×3 in the baseline model) leads to an accuracy drop of 0.58%. Reducing the number of output channels of the in-pixel convolution by 4× (8 channels from 32 channels in the baseline model), on the top of non-overlapping striding, reduces the test accuracy by 0.33%. Additionally, replacing the element-wise multiplication with the custom P^2M function in the convolution operation reduces the test accuracy by a total of 0.56% compared to the baseline model. Note that we can further compress the in-pixel output by either increasing the stride value (changing the kernel size proportionately for non-overlapping strides) or decreasing the number of channels. But both of these approaches reduce the VWW test accuracy significantly, as shown in Fig. 3.8(b).

Comparison with Prior Works: Table 4 compares different in-sensor and near-sensor computing works [12–15] in the literature with our proposed P^2M approach. However, most of these comparisons are qualitative in nature. This is because almost all these works have used toy datasets like MNIST, while some have used low-resolution datasets like CIFAR-10. A fair evaluation of in-pixel computing must be done on high-resolution images captured by modern camera sensors. To the best of our knowledge, this is the first paper to show in-pixel computing on a high-resolution dataset, such as VWW, with associated hardware-algorithm co-design. Moreover, compared to prior-works we implement more complex compute operations including analog convolution, batch-norm, and ReLU inside the pixel array. Additionally, most of the prior works use older technology node (such as 180 nm). Thus, due to major discrepancy in the use of technology nodes, unrealistic datasets for in-pixel computing, and only a sub-set of computations being implemented in prior-works it is infeasible to do a fair quantitative comparison between the present work and previous works in the literature. Nevertheless, Table 4 enumerates the key differences and compares the highlights of each work, which can help develop a good comparative understanding of in-pixel compute ability of our work compared to previous works.

Work	Tech Node	Computation	High Resolution	Dataset	Supported Ops.	Acc. (%)
P^2M (ours)	22 nm	Analog	Yes	VWW	Conv, BN, ReLU	89.90
TCAS-I 2020 [12]	180 nm	Analog	No	–	Binary Conv.	–
TCSVT 2022 [15]	180 nm	Analog	No	CIFAR-10	Conv.	89.6
Nature 2020 [13]	–	Analog	No	3-class alphabet	MLP	100
ECCV 2020 [14]	180 nm	Digital	No	MNIST	MLP	93.0

Table 3.4: Comparison of P^2M with related in-sensor and near-sensor computing works

Model type	Sensing (pJ) (e_{pix})	ADC (pJ) (e_{adc})	SoC comm. (pJ) (e_{com})	MAdds (pJ) (e_{mac})	Sensor output pixel (N_{pix})
P^2M (ours)	148	41.9	900	1.568	112×112×8
Baseline (C)	312	86.14			560×560×3
Baseline (NC)					

Table 3.5: Energy estimates for different hardware components. The energy values are measured for designs in 22nm CMOS technology. Note, the sensing energy includes the analog convolution energy for P^2M as analog convolution is performed as a part of the sensing operation. For the e_{mac}, we convert the corresponding value in 45nm to that of 22nm by following standard scaling strategy [1].

Notation	Description	Value
B_{IO}	I/O band-width	64
B_W	Weight representation bit-width	32
N_{bank}	Number of memory banks	4
N_{mult}	Number of multiplication units	175
T_{sens}	Sensor read delay	35.84 ms (P^2M) 39.2 ms (baseline)
T_{adc}	ADC operation delay	0.229 ms (P^2M) 4.58 ms (baseline)
t_{mult}	Time required to perform 1 mult. in SoC	5.48 ns
t_{read}	Time required to perform 1 read from SRAM in SoC	5.48 ns

Table 3.6: The description and values of the notations used for computation of delay. Note that we calculated the delay in 22nm technology for 32-bit read and MAdd operations by applying standard technology scaling rules initial values in 65nm technology [2]. We directly evaluated the T_{read} and T_{adc} through circuit simulations in 22nm technology node.

3.5.3 EDP Estimation

We develop a circuit-algorithm co-simulation framework to characterize the energy and delay of our baseline and P^2M-implemented VWW models. The total energy consumption for both these models can be partitioned into three major components: sensor (E_{sens}), sensor-to-SoC communication (E_{com}), and SoC energy (E_{soc}). Sensor energy can be further decomposed to pixel read-out (E_{pix}) and analog-to-digital conversion (ADC) cost (E_{adc}). E_{soc}, on the other hand, is primarily composed of the MAdd operations (E_{mac}) and parameter read (E_{read}) cost. Hence, the total energy can be approximated as:

$$E_{tot} \approx \underbrace{(e_{pix} + e_{adc}) * N_{pix}}_{E_{sens}} + \underbrace{e_{com} * N_{pix}}_{E_{com}} + \underbrace{e_{mac} * N_{mac}}_{E_{mac}} + \underbrace{e_{read} * N_{read}}_{E_{read}}. \tag{3.4}$$

Here, e_{sens} and e_{com} represents per-pixel sensing and communication energy, respectively. e_{mac} is the energy incurred in one MAC operation, e_{read} represents a parameter's read energy, and N_{pix} denotes the number of pixels communicated from sensor to SoC. For a

convolutional layer that takes an input $\mathbf{I} \in R^{h_i \times w_i \times c_i}$ and weight tensor „ $\in R^{k \times k \times c_i \times c_o}$ to produce output $\mathbf{O} \in R^{h_o \times w_o \times c_o}$, the N_{mac} [61] and N_{read} can be computed as,

$$N_{mac} = h_o * w_o * k^2 * c_i * c_o \tag{3.5}$$

$$N_{read} = k^2 * c_i * c_o \tag{3.6}$$

The energy values we have used to evaluate E_{tot} are presented in Table 3.5. While e_{pix} and e_{adc} are obtained from our circuit simulations, e_{com} is obtained from [16]. We ignore the value of E_{read} as it corresponds to only a small fraction ($<10^{-4}$) of the total energy, similar to [62–65]. Fig. 3.9(a) shows the comparison of energy costs for standard vs P^{2}M-implemented models. In particular, P^{2}M can yield an energy reduction of up to 7.81×. Moreover, the energy savings is larger when the feature map needs to be transferred from an edge device to the cloud for further processing, due to the high communication costs. Note, here we assumed two baseline scenarios one with compression and one without compression. The first baseline is MobileNetV2 which aggressively down-samples the input similar to P^{2}M ($h_i/w_i : 560 \longrightarrow h_o/w_o : 112$). For the second baseline model, we assumed standard first layer convolution kernels causing standard feature down-sampling ($h_i/w_i : 560 \longrightarrow h_o/w_o : 279$).

To evaluate the delay of the models we assume sequential execution of the layer operations [2, 66, 67] and compute a single convolutional layer delay as [2]

$$t_{conv} \approx \lceil \frac{(k)^2 c_i c_o}{(B_{IO}/B_W) N_{bank}} \rceil * t_{read} + \lceil \frac{(k)^2 c_i c_o}{N_{Mult}} \rceil h_o * w_o * t_{mult}. \tag{3.7}$$

where the notations of the parameters and their values are shown in Table 3.6. Based on this sequential assumption, the approximate compute delay for a single forward pass for our P^{2}M model can be given by

$$T_{delay} \approx T_{sens} + T_{adc} + T_{conv}. \tag{3.8}$$

Here, T_{sens} and T_{adc} correspond to the delay associated to the sensor read and ADC operation respectively. T_{conv} corresponds to the delay associated with all the convolutional layers where each layer's delay is computed by Eq. 3.7. Fig. 3.9(b) shows the comparison of delay

between P^2M and the corresponding baselines where the total delay is computed with the sequential sensing and SoC operation assumption. In particular, the proposed P^2M approach can yield an improved delay of up to 2.15×. Thus the total EDP advantage of P^2M can be up to 16.76×. On the other hand, even with the conservative assumption of total delay is estimated as $\max(T_{sens}+T_{adc}, T_{conv})$, the EDP advantage can be up to ~11×.

Since the channels are processed serially in our P^2M approach, the latency for the convolution operation increases linearly with the number of channels. With 64 output channels, the latency of the in-pixel convolution operation increases to 288.5 ms from 36.1 ms with 8 channels. On the other hand, the combined sensing and first layer convolution latency using classical approach increases only to 45.7 ms with 64 channels from 44ms with 8 channels. This is because the convolution delay constitutes a very small fraction of the total delay (sensing+ADC+convolution) in the classical approach. The break-even (number of channels beyond which in-pixel convolution is slower compared to classical convolution) happens at 10 channels. While the energy of the in-pixel convolution increases from 0.13 mJ with 8 channels to 1.0 mJ with 32 channels, the classical convolution energy increases from 1.31 mJ with 8 channels to 1.39 mJ with 64 channels. Hence, our proposed P2M approach consumes less energy than the classical approach even when the number of channels is increased to 64. That said, almost all of the state-of-the-art on-device computer vision architectures (e.g., MobileNet and its variants) with tight compute and memory budgets (typical for IoT applications) have no more than 8 output channels in the first layer [21, 49], which is similar to our algorithmic findings.

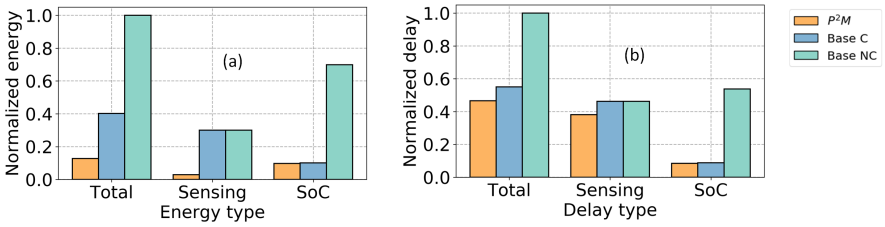


Figure 3.9: Comparison of normalized *total*, *sensing*, and *SoC* (a) energy cost and (b) delay between the P^2M, and baseline models architectures (compressed C, and non-compressed NC). Note, the normalization of each component was done by diving the corresponding energy (delay) value with the maximum total energy (delay) value of the three components.

3.6 Conclusions

With the increased availability of high-resolution image sensors, there has been a growing demand for energy-efficient on-device AI solutions. To mitigate the large amount of data transmission between the sensor and the on-device AI accelerator/processor, we propose a novel paradigm called *Processing-in-Pixel-in-Memory* (P^2M) which leverages advanced CMOS technologies to enable the pixel array to perform a wider range of complex operations, including many operations required by modern convolutional neural networks (CNN) pipelines, such as multi-channel, multi-bit convolution, BN and ReLU activation. Consequently, only the compressed meaningful data, for example after the first few layers of custom CNN processing, is transmitted downstream to the AI processor, significantly reducing the power consumption associated with the sensor ADC and required data transmission bandwidth. Our experimental results yield reduction of data rates after the sensor ADCs by up to $\sim 21\times$ compared to standard near-sensor processing solutions, significantly reducing the complexity of downstream processing. This, in fact, enables the use of relatively low-cost micro-controllers for many low-power embedded vision applications and unlocks a wide range of visual TinyML applications that require high resolution images for accuracy, but are bounded by compute and memory usage. We can also leverage P^2M for even more complex applications, where downstream processing can be implemented using existing near-sensor computing techniques that leverage advanced 2.5 and 3D integration technologies [68].

Chapter 4

FPCA: Field-Programmable Pixel Convolutional Array for Extreme-Edge Intelligence (FPCA)

4.1 Introduction

Given the proliferation of high-resolution and high-frame rate imaging, the surge in data generated by cameras for artificial intelligence (AI) enabled computer vision (CV) applications presents significant energy and bandwidth challenges. This issue is exacerbated by the conventional architecture that separates the sensor from the processing units, leading to inefficiencies in data transfer and computation [69, 70]. To address this, recent research has pivoted towards strategies that *process and compress* data closer to the point of capture, aiming to alleviate these bottlenecks.

To enhance efficiency in data processing and compression closer to the sensor, three primary methodologies have emerged, differentiated by how closely the processing unit is integrated with the data generation (sensor) unit:

1. Near-Sensor Processing: In this setup, the data processor is located in close proximity to the CMOS image sensor (CIS) chip. This arrangement boosts energy and bandwidth efficiency by reducing the distance between the sensor and the processor [71, 72]. Despite this, the processor and sensor remain on separate chips, meaning there is still a notable distance including significant off-chip communication that data must

traverse.

2. In-Sensor Processing: This method incorporates either an analog or digital signal processor directly into the periphery of the sensor chip. By doing so, it significantly reduces the physical distance between where data is generated and first processed [28]. This approach helps reduce the data transfer bottleneck between sensor and processor, yet the bottlenecks in transferring data from the sensor to the processor still exists as the sensor is physically separate from the processing unit that resides in the periphery outside sensor array.

3. In-Pixel Processing: Taking integration of sensor and processor a step further, this innovative strategy equips each pixel circuit within the sensor with the ability to perform massively parallel computations. This means processing can start at the very site of data capture, along each pixel row(s) and/or column(s), drastically diminishing the bandwidth requirement and reducing both the energy consumption and latency of the sensor-processor system.

Despite various efforts in in-pixel processing [29, 30, 73–76], most of the works fail to accomplish high accuracy in complex machine learning (ML) tasks. Complex ML tasks often require advanced operations like multi-bit, multi-channel convolution, batch normalization (BN), and Rectified Linear Units (ReLU). Most approaches [29, 30, 73, 74] have limitations, such as binary weight implementation and a lack of multi-channel convolution capability, while focusing on simpler datasets that do not fully represent the complexities of real-world CV applications. Notably, prior works [75, 76] have demonstrated significant improvements but are hindered by the fixed nature of the transistor-width-based weight implementation as well as the number of kernels and the stride size, which lacks the flexibility and reconfigurability needed for diverse CV applications. Thus, current research has failed to achieve the combination of high accuracy and reconfigurability for pixel sensors to tackle multiple CV applications within the same pixel array.

Furthermore, in-pixel processing involves co-design considerations across chip integration, circuit configurations, and algorithm. In previous attempts, embedding computational tasks like matrix-vector multiplication within the pixel array can compromise pixel density due to requirement of hundreds of added transistors per pixel based on size of stride and number of channels. Innovations such as heterogeneous 3D integration offer a pathway to

vertically stack logic or memory substrates with the CIS [77, 78], potentially overcoming these limitations. However, the pixel pitch limits the number of transistors which in turn limits the size of stride, channels and kernel size of an neural network which effects the overall accuracy of the CV task.

This work introduces, for the first time, a novel Field-Programmable Pixel Convolutional Array (FPCA) pixel architecture design for in-pixel processing by enabling comprehensive field-programmability of all the key parameters within the initial layers of modern convolutional deep learning networks. Leveraging a hybrid CMOS-NVM circuit, this system introduces a reconfigurable structure designed to dynamically adjust all key parameters for state-of-the-art deep learning networks:

1. **Weight values**: allowing for modifications based on varying algorithm demands.
2. **Channel configurations**: adapting to different data bandwidth and processing needs.
3. **Kernel sizes**: customizable to match the specific convolutional requirements of different applications.
4. **Stride sizes**: offering flexibility in feature sampling and data throughput.

Such reconfigurability is crucial for creating *sustainable solutions*, wherein a single hardware setup can meet a wide range of deep learning algorithm specifications. For instance, complex datasets like BDD100K [79] might require smaller filter sizes and strides to capture details, while simpler datasets like visual wake word (VWW) [54] can work with larger kernels and broader strides while maintaining classification accuracy. By incorporating these adjustable parameters, the FPCA not only addresses the static nature of previous systems but also significantly enhances the scope and application space for in-pixel processing use cases. This adaptability leads to field-programmability, ensuring that the same physical infrastructure can continuously evolve in response to emerging computational challenges and advancements in machine learning methodologies involving convolutional operations. Moreover, our approach significantly reduces the required area per pixel by eliminating the need for weight transistors within each pixel unit and placing the weight block composed of non-volatile memory (NVM) in a separate weight die using 3D integration technology and shared with pixels using modified rolling shutter operation. This advancement not only enhances the density and scalability of the pixel array but also opens up new possibilities

for deploying more complex neural network models directly onto the hardware. Through detailed simulation results on TSMC 28nm, we demonstrate the effectiveness of our design in performing dot product operations— a fundamental building block of neural network computations— with improved efficiency and adaptability compared to existing designs. Furthermore, we present a novel machine learning framework compatible bucket-select curvefit approach to accurately model the non-linearity associated with analog convolution operation.

Also, the proposed FPCA scheme offers distinct advantages over existing near-sensor and in-sensor processing approaches, particularly in energy-constrained applications like autonomous drones and smart glasses. While near-sensor processing incorporates ML accelerators on the same PCB or in 3D-stacked configurations with the CIS chip, and in-sensor processing integrates processing unit within the CIS chip periphery circuit, both approaches still face fundamental data transfer limitations. Near-sensor processing must handle the energy overhead of data transfer between the CIS and processing chip, while in-sensor processing requires sequential streaming of data from photodiode arrays to peripheral circuits. In contrast, FPCA improves the camera's energy-delay product by enabling computation within the pixel array, thereby significantly mitigating data transfer to/from the pixel array, while also reducing the number of times ADC conversions are needed. Thus, in contrast to near- and in-sensor processing schemes, FPCA addresses data movement challenges at the level of the pixel read-out circuit. Finally, FPCA incorporates reconfigurability for all key parameters of the deep learning network inside the proposed hardware, thereby making it much more versatile compared to prior works [75, 78, 80]. Importantly, FPCA is not meant to replace these existing techniques but can complement them – it can be integrated with both near-sensor and in-sensor processing schemes to provide additional benefits.

For this paper in section 4.2 we discuss previous works that are related to in-pixel computing, section 4.3 introduces the novel NVM in-pixel computing circuit and the reconfigurability for all key ML parameters including weight value, kernel size, stride, channels, pixel skipping and demonstrate a novel bucket select curvefit approach to capture the non-linearity in the analog output voltage suitable for incorporation in standard ML training frameworks. Section 4.4 introduces a new modeling method for FPCA circuit analog convolution output. Then section 4.5 will elaborate on the simulation results and the analysis of the FPCA pixel circuit. The last section, section 4.7 concludes the paper.

4.2 Background and Related Work

In the first layer of convolutional neural networks (CNNs), the initial processing involves the multiplication of pixel outputs from the camera sensor with multi-bit weight values, a critical step in data interpretation and analysis [81]. To facilitate this within the pixel array without compromising the resolution of the CIS, previous approaches [75, 76, 78, 82] embed these weights directly into the pixel architecture. This embedding is made possible through the use of advanced 3D integration technologies, which allow for the vertical stacking of weights [83, 84]. The physical implementation of these weights can be achieved through adjusting CMOS transistor geometry or by leveraging the resistance states in various non-volatile memory (NVM) devices such as Resistive Random Access Memory (RRAM), Phase Change Memory (PCM), and Magnetic Random Access Memory (MRAM) [85].

Additionally, the algorithm necessitates use of both positive and negative weight values to maintain accuracy in the test phase, requiring innovative circuit techniques to process and distinguish these weights accurately. In [75], a novel use of the peripheral Single-Slope (SS) Analog-to-Digital Converter (ADC) is proposed. This approach combines the results of positive and negative weights by respectively increasing and decreasing the counter inside the ADC, enabling the calculation of the final convolution output. The integration of non-linear activation functions, such as the Rectified Linear Unit (ReLU), is ingeniously achieved by repurposing the on-chip Correlated Double Sampling (CDS) circuit found in CIS alongside the SS ADC. This setup ensures that the final ADC count, post CDS operation (which includes both ‘up’ and ‘down’ counting), results in a non-negative value, thereby effectively implementing the ReLU operation. Our proposed FPCA architecture also leverages this approach for the periphery ADC circuit design.

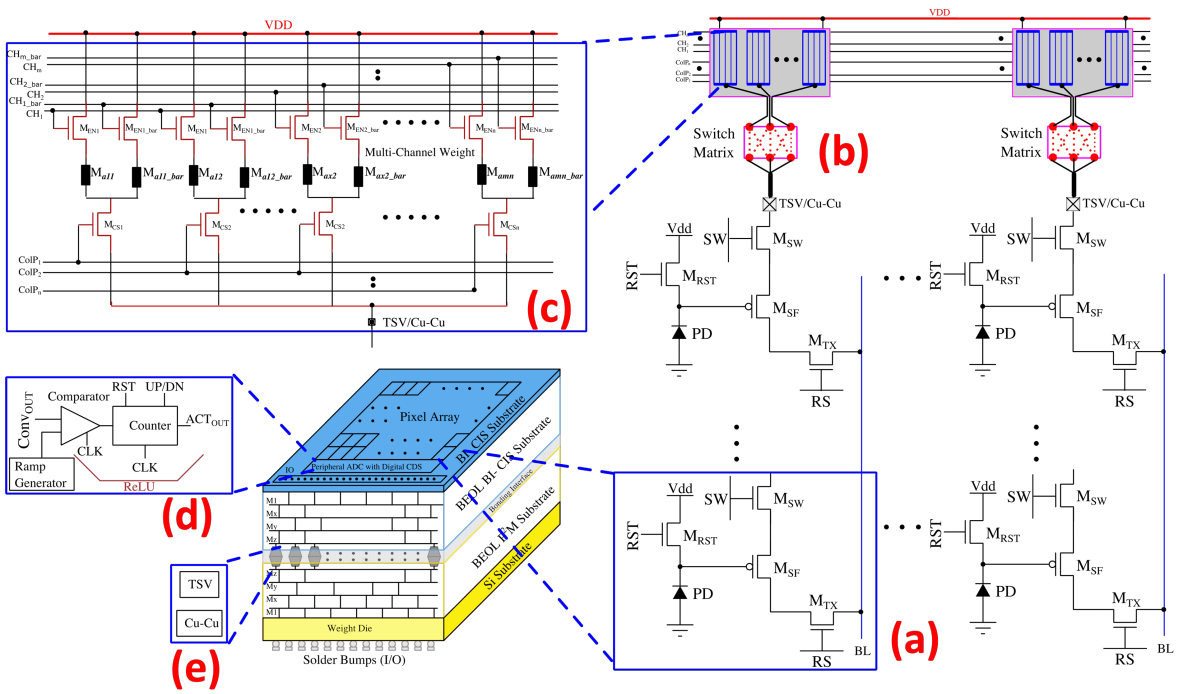


Figure 4.1: Proposed circuit and the overall architecture of the FPCA, where (a) is the novel 4T APS schematic, (b) is the switch matrix that connects the pixel array to the shared weight block in a 3D integrated weight die, (c) is the example diagram of shared weight block, (d) is peripheral ADC and (e) is the connection between the two dies using either Through-Silicon Vias (TSV) or Copper-Copper bonding (Cu-Cu).

Moreover, the integration of the batch normalization (BN) layer, crucial for training convergence, is partially combined with both the convolutional and ReLU layers. This is implemented by initializing the counter with the BN offset term and adjusting the weights with the BN scale term [75]. The processed activations are then ready to be transmitted off-chip via various I/O technologies such as low voltage differential signaling (LVDS), interposer (2.5D integration), through-silicon via (TSV), Cu-Cu bonding and Wireless, among others. [77, 78]

4.3 FPCA Architecture and Reconfigurability

The essential innovation of the FPCA architecture is the reconfigurability it provides on different levels of the circuit. As shown in Fig. 4.1, the proposed new circuit is composed of

novel 4-transistor unit pixel circuit (Fig. 4.1(a)), switch matrices (Fig .4.1(b)), multi-channel weight block for each pixel column (Fig. 4.1(c)), and periphery ADC (Fig. 4.1(d)). The pixel array and the multi-channel weight array are on separate dies that is connected by through-silicon vias (TSV) or Cu-Cu hybrid bonding using 3D integration (Fig. 4.1(e)) [77]. These different blocks provide the architecture the reconfigurablity in all the key ML parameters: weight value, kernel size, channel size, stride size and the ability to achieve region skipping for neural network algorithms. For each column, it has one multi-channel weight block within the weight array that stores all the channels' kernel weight values using non-volatile memory (NVM). A weight value is represented by two NVMs (M_{a11}&M_{a11_bar}), one to represent the positive value and one for negative. If a weight value is negative then the NVM for positive value M_{a11} stores 0 and the negative value is stored in the M_{a11_bar}, the NVMs is connected to the source of transistor M_{EN}&M_{EN_bar} that is used for selecting one channel in the output feature map. During a specific convolution operation, multiple pixels are activated and only one of the channels is enabled. Convolutions for multiple channels are performed sequentially. For each channel, the NVM for the positive weight is connected to the channel line CH_i and the negative weight to CH_{i_bar}, these two lines are activated sequentially, for positive & negative signal accumulation. Further, each NVM storing a weight value is also connected to a kernel column select transistor M_{CSi}. The gate of these column select transistors are controlled by the input of Column Pattern control line $ColP$. The source of the M_{CSi}s are then connected to the switch matrix. The output of the switch matrix would then connect to the drain of the transistor M_{SW} within each unit pixel circuit. We will detail on these circuit structures and their interconnections in the following paragraphs.

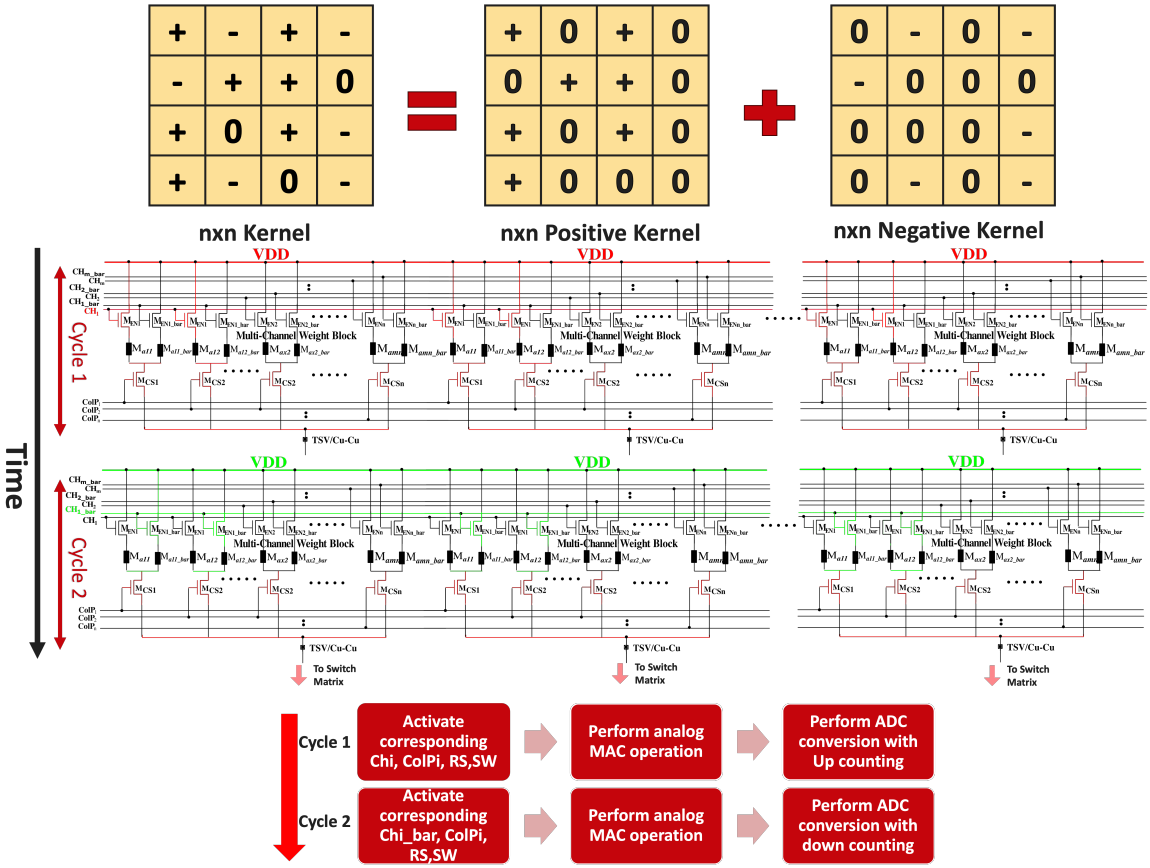


Figure 4.2: Detailed Multi-channel Weight Block (shared weight block) Schematic with positive and negative kernel. The top part of the figure is the representation of the proposed method to store the kernel weight using one positive and one negative kernel, and the middle part of the figure is the schematic representation of the two cycles of the multi-channel weight block to show the activated part of the circuit.The bottom part is the sequence diagram for a complete convolution computation.

4.3.1 In-situ Multi-pixel Convolution Operation

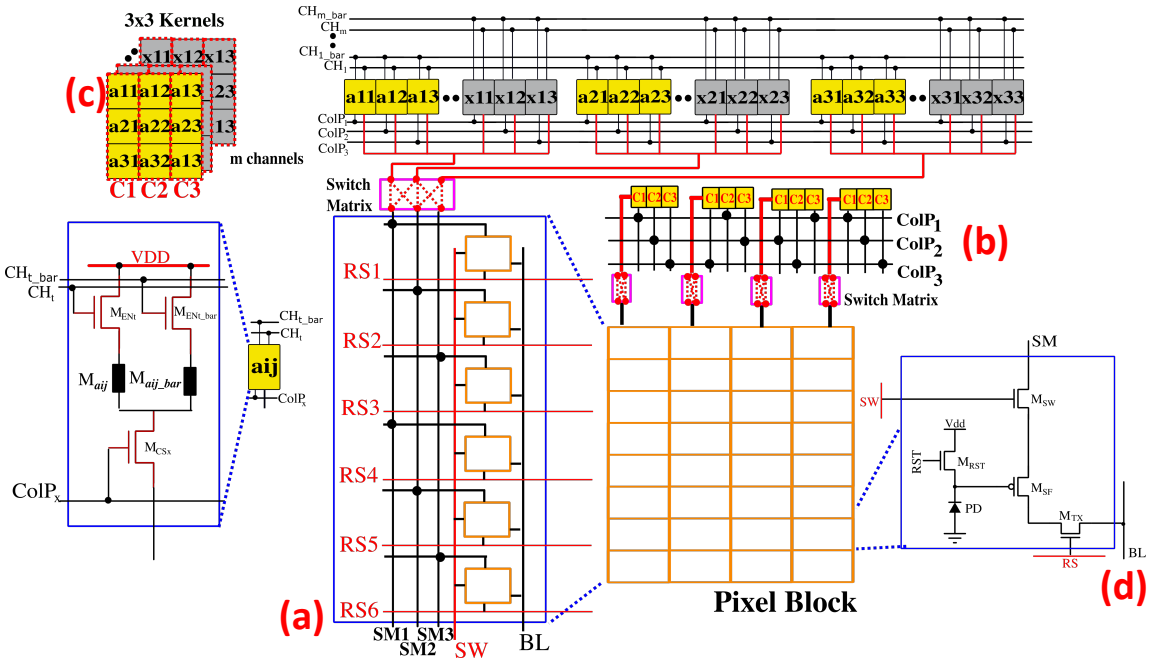


Figure 4.3: Detail architecture of the column design of FPCA pixel array and Multi-Channel Wight Block where (a) is pixel column design, (b) shows the connection pattern to different columns of the multi-channel weight block and the control signal ColP (column pattern select line), (c) is example figure of a m-channel with max 3×3 kernel and (d) is the pixel circuit design where the SW line is the column control line and RS is the row control line and the input to the pixel: line SM is connected to one node of the switch matrix.

To achieve the convolution operation, we simultaneously activate multiple pixels. For example, we activate $n_X \times n_Y \times 3$ pixels at the same time, where n_X and n_Y denote the spatial dimensions and 3 corresponds to the RGB (red, blue, green) channels in the input activation layer. For each activated pixels, the pixel output is modulated by the photo-diode current and the weight value of the activated $M_{a_{ij}}$ NVM in the multi-channel weight block shown as Fig. 4.1(c) associated with the pixel. In the FPCA architecture, the unit pixel circuit schematic follows up on previous works [75–78]. For each pixel in the FPCA pixel array, the output voltage approximates the multiplication of light intensity and corresponding weight in the multi-channel weight array. For each output bit line, shown as vertical blue lines BL in Fig. 4.1(a), the cumulative pull up strength of the activated pixels connected to that line drives it high. The increase in pixel output voltages accumulate on the bit lines BL

implementing an analog summation operation. Consequently, the voltage at the output of the bit lines represent the convolution operation between input activations and the stored weight in the weight die.

In summary, the presented unit pixel circuit can perform in-situ multi-bit, multi-channel analog convolution operation inside the FPCA pixel array, wherein the input activations are within the individual pixel (photodiode current) and the network weights are present in a shared weight block along with associated metal interconnects in the separate weight die connected to the pixel chip using TSV or Cu-Cu bonding [77, 78].

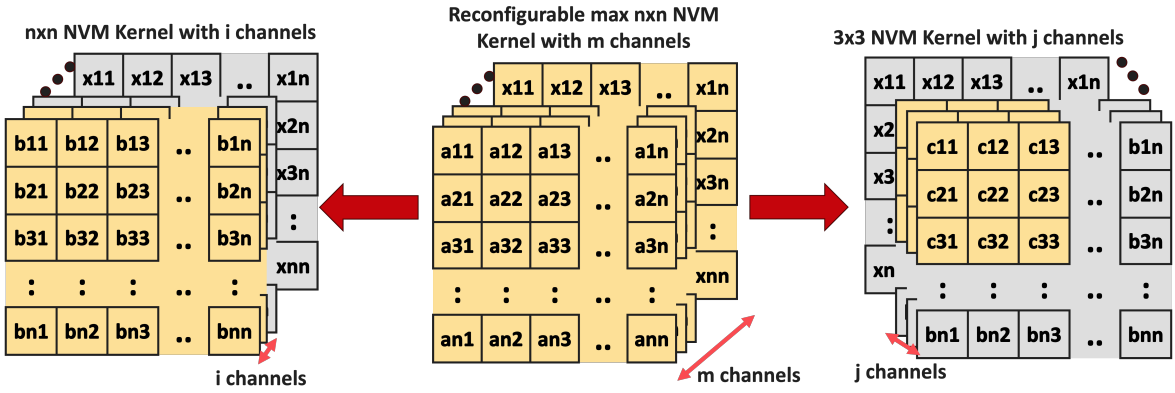


Figure 4.4: Reconfigurability in weight value, kernel size and channel size, the center of the figure is representing m channels of $k \times k$ kernels, the left part shows reconfigured weights in i channels of $k \times k$ kernels and the right part of the figure shows the reconfigured j channels for smaller 3×3 kernels.

4.3.2 Shared-Weight Block for In-pixel Convolution Operation

In CNNs, weights can take on both positive and negative values. However, as the NVM weights used in FPCA are inherently non-negative, direct representation of negative weights is not feasible. To address this, we leverage the on-chip digital Correlated Double Sampling (CDS) circuit, as shown in Fig. 4.1(d), commonly integrated into state-of-the-art commercial CMOS Image Sensors (CIS). This circuit is typically used alongside column-parallel Single Slope ADCs (SS-ADCs) to eliminate reset noise during analog-to-digital conversion. We examined training deep learning networks under a "positive-only" weight constraint and observed that, while acceptable performance can be maintained for simple datasets such

as MNIST, accuracy drops dramatically for more complex datasets. This finding indicates that supporting signed weight representation is essential for realistic CNN workloads. Therefore, FPCA adopts a dual-path approach that allows both positive and negative weight computation within each convolution channel. An SS-ADC operates by comparing an input analog voltage to a ramping voltage with a fixed slope generated by a ramp generator. A comparator monitors this ramp, and a counter increments until the ramp voltage surpasses the input voltage. At this point, the counter's output is latched, providing the digital equivalent of the input analog voltage. The digital CDS circuit further processes two correlated samples: the first captures pixel reset noise, and the second records the actual signal combined with reset noise. By taking the difference between these two samples, the CDS effectively cancels out the reset noise. We repurpose the noise-canceling functionality of the digital CDS circuit to represent both positive and negative weights. Each weight NVM is designated as either positive or negative by connecting it to distinct control transistor—CH_i for positive weights or CH_ibar for negative weights. As depicted in Fig. 4.2. For each channel, multiple pixels are activated to compute the inner product, and two samples are read sequentially.

For the weight representation, as shown in Fig. 4.2, a $n \times n$ kernel with both positive and negative weight values can be treated as two $n \times n$ kernels, one that stores only positive weight values and the other that stores the negative values. For negative weight values in the original kernel, a zero is stored in the corresponding position within the positive kernel. Conversely, where the original kernel has positive weights, zeros are placed in the corresponding positions of the negative kernel. This scheme requires a total of 2 cycles for 1 in-pixel convolution operation. As shown in the bottom part of Fig. 4.2, for the first cycle, the CH_i line would be pulled up to VDD to activate the positive weight values within the kernel (as shown by the red lines in Fig. 4.2). And in the second cycle the CH_{i_bar} line would be pulled up while deactivating others to activate the negative weight through its corresponding M_{EN_bar} transistor, as shown by green lines in Fig. 4.2. The activated NVMs in each cycle would then connect to the switch matrix and then to a certain pixel within the pixel array. Note that for a single channel, all positive weights' M_{ENi} transistors' gate are connected to the same channel select line CHi, and all negative weights' M_{ENi_bar} transistors are connected to the complement channel select line CHi_bar. Therefore for a convolution operation:

1. In the first step, the “red lines” (positive weights) control gates CH_i and $ColP_i$ are supplied with a VDD, while the CH_ibar (negative weights) is grounded. The mapped weights are sent to the pixel array and The resulting accumulated dot product is digitized by the SS-ADC in the output of BL (Fig. 4.1(a)) as the counter performs an "up" count.

2. In the second step, the “green lines” (negative weights)’s gates CH_ibar and $ColP_i$ are supplied with a high VDD, while the CH_i is grounded. This accumulated dot product is also digitized, and its value is subtracted from the first sample using the SS-ADC, with the counter performing a "down" count.

This approach ensures that the digital CDS circuit first accumulates the convolution results for all positive weights, then subtracts those for all negative weights, effectively implementing the signed inner product for each channel. Channel-specific select lines govern this operation.

Note, a typical RGB camera has 3 input channels. Our system allows these three color channels to operate concurrently by sharing channel select lines. If we have a total of c_o output channels, the maximum kernel size across all output channels is $n \times n \times 3$, where the 3 corresponds to the RGB channels being processed together. This simultaneous activation across the 3 input channels ensure that the RGB channels are processed simultaneously and their convolution outputs are accumulated together in accordance with the algorithmic requirement for modern CNNs. Consequently, the FPCA architecture would have $2 \times c_o$ channel select lines (CHis and CHi_bars), and each column of the pixel array would have in total $2 \times n^2 \times 3 \times c_o$ number of weights per pixel column that is on a separate weight die. Note, this scheme allows the weights to be shared along columns, significantly reducing the number of transistors for representing weight kernels.

4.3.3 Mapping of Weights in Shared Weight Block with the Pixel Array for Convolution Operation

As described earlier, Channel Select Line CHi selects weight transistors M_{CSi}, similarly Column Pattern Select Line ColPi and the switch matrix control the mapping of the weight block to a certain pixel column. Fig. 4.2 shows the detail schematic for the connection of the multi-channel weight block and the channel select transistors M_{EN}s, in addition Fig. 4.3

shows the detailed circuit connection of the rest of shared weight block and the pixel column including switch matrix. As shown in Fig. 4.3(c), for an example 3×3 kernel the different rows (row1: $a11, a12, a13$ etc., row2: $a21, a22, a23$ etc.) are connected to the different input nodes of the switch matrix, the $a11, a12, a13$ that belong to the first row are connected to the first red node of the switch matrix while $a21, a22, a23$ are all connected to the second node. The ColP lines (as shown in Fig. 4.3(b)) are being used in order to select a specific column within a kernel for mapping the weight to the corresponding pixel column. For example, if ColP1 is being pulled up, C1 ($a11, a21, a31$) is connected to column 1 of pixel block, C2 ($a12, a22, a32$) is connected to column 2 of the pixel block. Each different ColP line enables assigning a different column of the kernel to a specific pixel column. Therefore, with the maximum kernel size of $n \times n$, there would be a total of n lines of ColPs for column pattern selecting (as one pixel column could be assigned to every column of the weight kernel). This pattern ensures that any column of the kernel could be mapped to a specific pixel column.

The bottom n nodes of the column switch matrix connect to the n lines of SM that is connected to the transistor M_{SW} within each pixel. For example, in Fig. 4.3(a), there are 3 SM lines. Note that the SM lines are routed in 3D integrated chip and these lines would incur no area overhead for the pixel array. Each pixel would connect to one SM line. Pixels in row 1 connect to the $SM1$, pixel in row 2 connects to $SM2$ and row n connects to the SMn, then for the row n+1, it would connect to the $SM1$, every next row would follow the same pattern. Therefore, through this design, it is made sure that every nearby column-wise k pixels (as the maximum kernel size would be $n \times n$) would be connected to a separate SM line that links to a unique node within the switch matrix. The switch matrix is configured to route these nn separate SM lines to correspond to the n different weights in the specific column of the kernel.

As for the control signals, as shown in Fig. 4.3(d), within each column pixel there are two control lines, shown in red, RS and SW, used for horizontal and vertical enabling the unit pixel. The SW line is used for column enabling and the RS line is used for row enabling. When both RS & SW lines are enabled simultaneously the output of the corresponding unit pixel is connected for read operation. A more general diagram of the mechanism is shown in Fig. 4.3(a). Each row shares the same RS control line while every SW line are shared among all pixel units within a column.

Therefore, for the overall matrix multiplication operation for each FPCA pixel column

with k × k kernel: channel select line CHi and the column pattern select line ColPi enables the specific k weights in the k × k kernel to be connected to the n nodes in switch matrix; and the switch matrix controls the connecting pattern of the n SM lines that connect to the pixels in the pixel array.

4.3.4 FPCA Reconfigurability

Leveraging the circuit structures described above, the proposed FPCA architecture offers seamless reconfigurability on different levels, here we will discuss each level thoroughly.

4.3.4.1 Reconfigurable Kernel Size

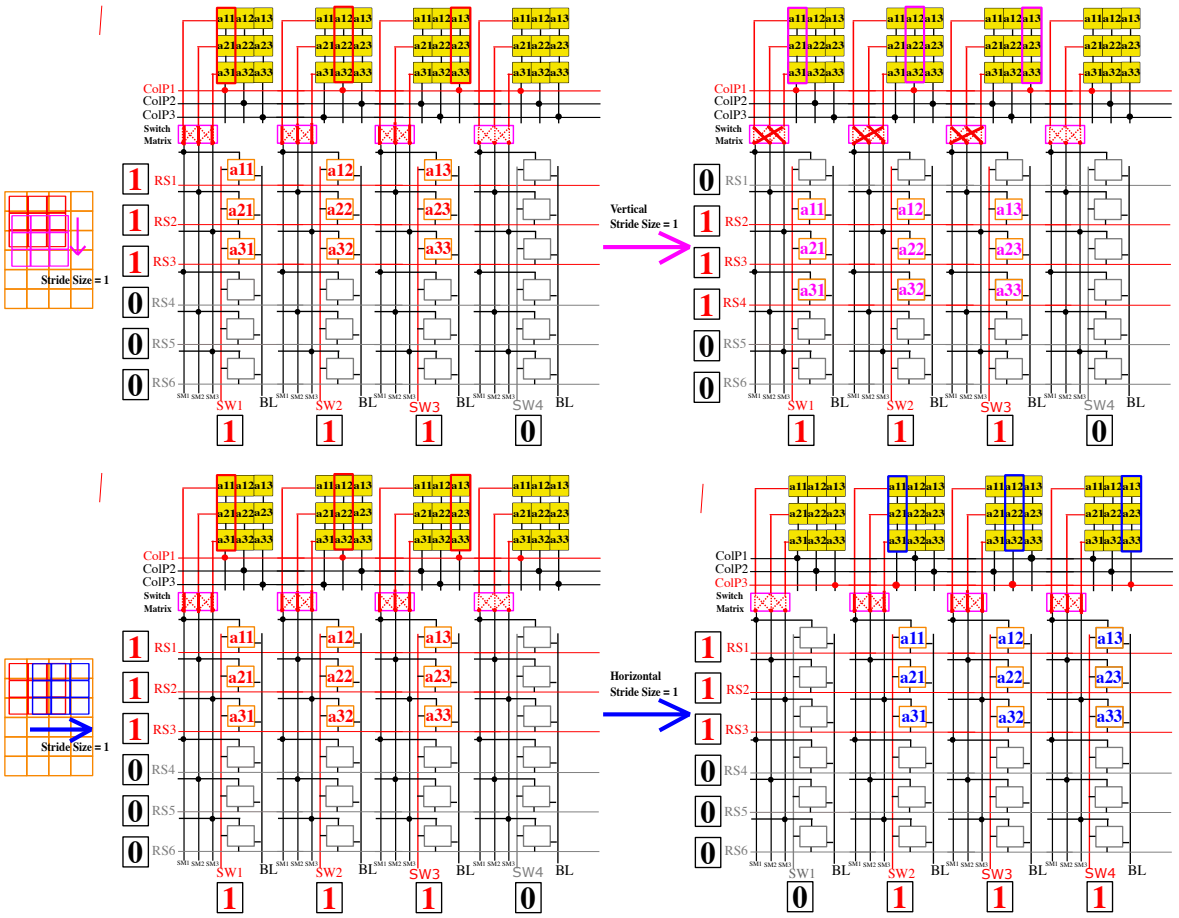


Figure 4.5: Figure representing vertical and horizontal striding of size s = 1 within the FPCA array.

Given the absence of a direct control mechanism to selectively deactivate segments of the kernel within the shared weight block external to the pixel array, our system invariably maps the entire kernel into the pixel array for each convolution operation. To circumvent this limitation and to introduce kernel size reconfigurability, our design employs the strategy of writing 0 weight values to multi-channel weight block, as is illustrated in Figure 4.4:

In our design, we have established a predetermined maximum kernel size of $n \times n$ for each channel. To accommodate arbitrary kernel sizes smaller than the maximum, our approach involves assigning zero weight values to the slots not utilized within the maximum kernel configuration. This technique leverages the pre-loading of kernel weight values before the convolution operation for inference-only task, thus adding no overhead during compute operation. It is important to note that with this method, the output bit line (BL) of each column in the pixel array will consistently reflect the activation of a fixed number of pixels, corresponding to the maximum column size n for a kernel size of $n \times n$. This ensures uniform number of pixels are activated, irrespective of the actual kernel size that may vary based on specific CV application.

In addition, the implementation of a logical zero-weight state is naturally achieved by programming the corresponding NVM cell into its high-resistance state. This approach is widely adopted in NVM-based convolution engines [86–88], where the high-resistance state contributes negligibly to the accumulated photocurrent during convolution. Our prior P2M architecture has also demonstrated that representing zero weights through high-resistance values results in no measurable accuracy degradation at the algorithm level [75, 76, 80], since the effective current contribution is much smaller than that of programmed weight states. The wide conductance dynamic range available in NVMs like RRAM further improves the robustness of this representation compared to transistor-based analog weights, allowing FPCA to map unused kernel elements cleanly and without additional circuit overhead.

Note that a detailed modeling of the NVM write process is not included in this work, as FPCA is designed for inference only operation where synaptic weights are programmed only once during initialization or occasional reconfiguration. Therefore, write latency and transient programming behavior do not affect runtime performance. In practical NVM technologies such as RRAM, multi-level programming is typically achieved using iterative write–verify schemes [86, 87], where each write pulse is followed by a verification step to ensure the conductance target is reached. These closed-loop methods have been widely demonstrated to reliably achieve multi-bit precision for inference-oriented systems, and

thus provide a suitable programming mechanism for FPCA without requiring low-level modeling of the write dynamics.

4.3.4.2 Reconfigurable Channel Size

Shown in Fig. 4.4, as each column of the pixel array connects to the multi-channel shared weight block that stores weights for all the c_o channels of maximum $n \times n$ kernel size, to reconfigure the channel size, we can simply control which CH line (channel select) line to activate as shown in Fig. 4.1(c).

4.3.4.3 Reconfigurable Stride Size

Our proposed FPCA structure allows any stride size ranging from 1 to n, where n is the maximum kernel column or row number. Fig. 4.5 is the example schematic for the FPCA implementing both vertical stride and horizontal stride size of minimum size $s = 1$. For the operation, when striding vertically, the column pattern control signal (ColP) stays the same when the kernel moves down as the kernel is still mapping to the same pixel array columns. By rerouting using the switch matrix, the different SM lines can be reorganized to connect to different columns of the kernel. For example, as represented in Fig. 4.5, switch matrix is rerouted to allow vertical striding in upper half of the figure. The SW lines are further activated in accordance with the kernel size being mapped in the pixel arrays. As the kernel strides vertically, different RS lines are pulled up in each cycle, while the SW lines stay the same. For reconfiguring vertical stride size, the activation scheme for RS lines changes according to the desired stride size, while also reconfiguring the switching matrix.

For horizontal stride, Fig. 4.5 shows the working schematic for horizontal stride size $s = 1$. When the kernel moves horizontally, the switch matrix routing would stay the same, the column pattern control line for ColPs would be selected in accordance with the stride size. For example, for horizontal stride size of 1, ColP1 activation is followed by ColP3 activation. This ensures when ColP1 is active, the first column of the kernel is mapped to the first column of the pixel array, while when ColP2 is active the second column of the kernel is mapped to the first column of the pixel array, implementing the horizontal striding. Meanwhile to select different columns different groups of the SW lines would be ON.

This configuration is essential for allowing the weight kernel stride horizontally in the pixel array, the total of n ColP lines (maximum kernel size is $n \times n$) allows the kernel

to move to every location in the pixel array horizontally. Note that Fig. 4.5 represents single kernel convolution operation, the FPCA structure allows massive parallel convolution operations along a set of selected rows in the pixel array.

4.3.4.4 Multi-Cycle Convolutional Operations

As mentioned in section 4.3.4.1, for any kernel size that is smaller than the maximum $n \times n$, 0 weight values would be written into the NVMs that are not occupied in the predetermined maximum kernel for each channel. Therefore, kernels of size $n \times n$ are applied to the pixel array, although some weights are set to hold 0 value. But as the stride size vary, it would require different number of cycles to complete the convolution operation for all stride locations. For a kernel with n for the maximum kernel width of one channel with a depth of 1, the total cycle needed for a stride size of S is $\frac{lcm[S,n]}{S}$. Thus, the total cycle number N_C needed for generating the output kernels for the next layer of convolution would be

$$N_C = 2 \times h_o \times c_o \times \frac{lcm[S,n]}{S} \tag{4.1}$$

where h_o is the height of the output kernel, c_o is the number of output channels, S is the stride size, n is the maximum kernel width.

4.3.4.5 Pixel Region Skipping

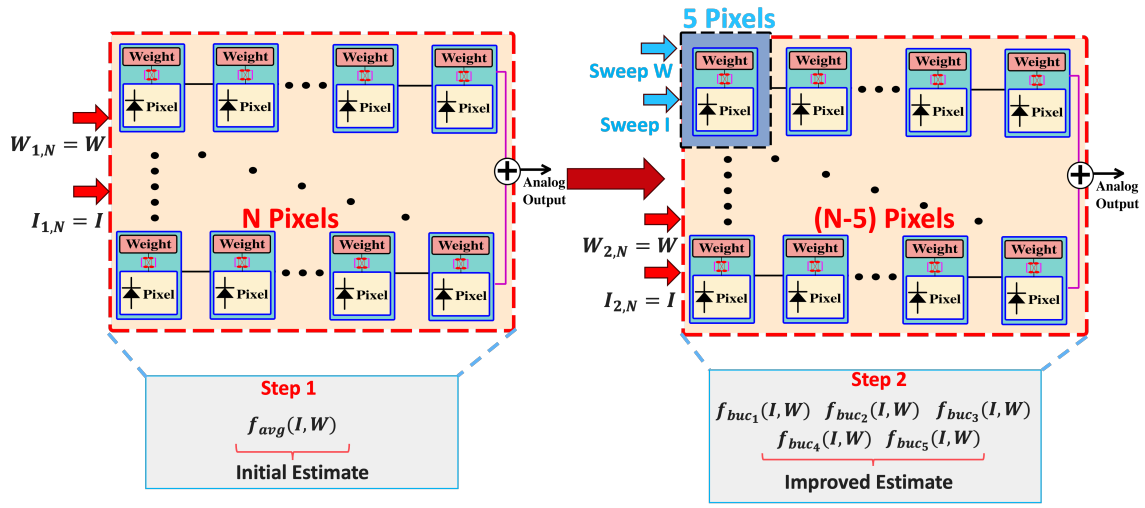
The FPCA architecture incorporates the region skipping functionality, achieved by the management of row-wise (RS) and column-wise (SW) control lines as depicted in Fig. 4.5. To enable unit-wise control over which specific rows and columns of the pixel array are activated, a total of $R_P \times C_P$ number of SRAMs along the periphery of the pixel array circuit would be needed. Here, R_P and C_P represent the total number of rows and columns in the pixel array, respectively. These SRAMs are intended to store control data for the RS and SW lines. However, $R_P \times C_P$ number of SRAMs is considerably high. Such an approach would result in substantial area overhead. Consequently, a block-wise approach to region skipping is recommended. This method reduces the complexity and number of SRAMs needed by grouping pixels into blocks and storing a single set of RS and SW values for each block. This strategy not only minimizes the area overhead but also simplifies the control scheme, making it a more practical solution for efficiently managing region

skipping within the FPCA framework. For example, if the entire pixel array is divided into 8x8 blocks, then only 64 SRAM locations are needed to hold the region skipping data.To identify the region of interest (ROI), several approaches can be employed, ranging from computer vision techniques to machine learning methods. Traditional computer vision approaches include edge detection using techniques like Canny edge detection [89], which identifies boundary discontinuities [90], and can separate regions based on pixel intensity values, while color-based segmentation can isolate regions with specific color characteristics [91]. More sophisticated deep learning approaches employ CNNs for automated feature learning and region detection, with architectures like YOLO (You Only Look Once) [92] and R-CNN (Region-based CNN) [93] providing robust ROI detection capabilities. These neural network-based methods can learn complex patterns and features from training data, enabling more accurate and adaptable ROI detection compared to traditional methods. The choice of ROI detection method depends on factors such as the specific application requirements, computational resources available, and the characteristics of the target regions to be identified. In this workflow, data processed by the FPCA chip is transmitted to the backend processor, where these methods are applied. After identifying the ROI, the information is fed back to the FPCA based pixel array to adjust control signals for the subsequent image capture.

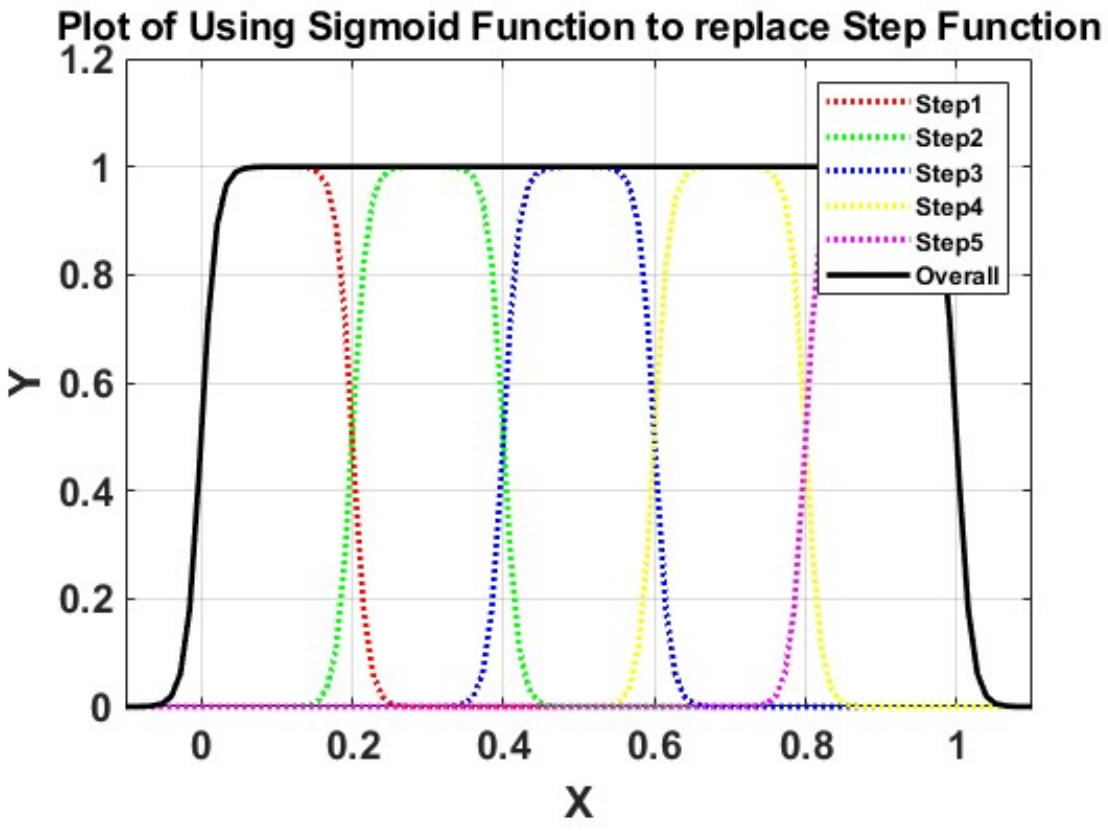
4.4 Accurate Modeling of Analog Convolution through Bucket-Select Curvefit Function

Analog computing including the proposed FPCA system shows inherent non-linearity due to non-linear behavior of constituent devices including NVM and transistors. This non-linear behavior needs to be accurately modeled in the algorithmic framework to mitigate any accuracy loss [76, 79]. However, a significant challenge arises as each pixel's output is influenced by the cumulative operation of other pixels that are connected together and activated simultaneously to perform parallel dot product operation. This inter-dependence of a given pixel's behavior on other pixels complicates accurate and machine learning framework compatible modeling of analog computing performed in the FPCA. Note that detailed device-level variability is not included in the present FPCA model, as the goal of this work is system-level architectural demonstration rather than low-level device charac-

terization. In practice, NVM-based accelerators commonly exhibit conductance variation and programming inaccuracy, but prior RRAM- and PCM-based deep learning studies have shown that such variations can be effectively handled through algorithm–hardware co-training [86, 94, 95]. The proposed bucket-select curvefit model captures the aggregate analog transfer behavior of the NVM–CMOS–interconnect pipeline, allowing the ML training framework to internalize these system-level non-idealities without requiring explicit device physics modeling.



(a)



(b)

Figure 4.6: Conceptual figure showing modeling approach of the proposed bucket select curvefit function. (a) is the diagram for the two steps of the novel curvefit bucket selection function. (b) is the figure depicting use of sigmoid functions to replace step functions.

We propose a novel two step bucket-select curvefit method to accurately model non-linear behavior associated with the analog computing FPCA system. Fig. 4.6(a) shows the conceptual diagram representing step 1 and 2 of the proposed bucket select curvefit method. In step 1, an initial estimate for the analog output voltage is obtained by using a *generic* analytical function. This generic analytical function is obtained by curve fitting a 2D surface plot to the analog SPICE output of N pixels, where input current to all the N pixels are kept the same and swept within a range of minimum and maximum current values. Similarly, the weights associated with all the N pixels are also kept the same, and swept within a range of minimum and maximum weight values. The analog output estimate obtained from this step is used as an input to step 2. Thus, a total of 2N parameters (N input current and N weights) along with the initial estimate of analog voltage serves as input to step 2.

The initial estimate categorizes the output into one of several predetermined smaller ranges, referred to as 'buckets'. Each bucket, representing a segment of the total output range, is associated with a unique curvefit function. Thus, for each range the analog behavior is modeled by a curvefit function specifically tailored to the range of interest as determined by the initial estimate. For example, the overall analog voltage range from 0V to 1V can be sub-divided into 5 different buckets each of 200mV size. The initial estimate obtained in step 1 helps select the bucket (or range) of interest. For each bucket, a specific curvefit function models the behavior of the pixels (or the error associated with the initial estimate) accurately in the vicinity of the voltage range associated with the bucket. The analog output voltage predicted by step 2 is therefore, much more accurate compared to the initial estimate. This is because although the behavior of any pixel is dependent on the cumulative effect of other pixels it is connected to, yet the pixel's response is a strong function of its own input current and weight and only a weak function of the cumulative effect of other pixels. Thus, step 1 helps to estimate the cumulative effect of the pixels using a generic curvefit function, while step 2 uses a tailored curvefit function to accurately model the strong dependence of a pixel on its input current and its weight based on the initial estimate obtained from step 1.

The bucket curvefit functions are obtained by modifying the simulation setup of step 1. Majority of the pixels still share same value of input currents and weights. The value is chosen such that the output is forced to be within the range of a specific bucket, mimicking the cumulative effect of the interconnected pixels. A small subset of pixels then undergo a parameter sweep, leading to the generation of a distinctive curvefit function specific to the bucket of interest.

For our simulations, we considered a kernel size of $5 \times 5 \times 3$, the overall model consists of a total of 6 curvefit functions: the first one is the generic function: f_{avg}, it is obtained by sweeping I (Input current or light intensity) and W (Weight) wherein all 75 pixels share the same parameter values for input current and weights. The rest 5 are the bucket curvefit functions f_{buc_1} to f_{buc_5}, they are generated by sweeping a small subset of 5 pixels' I and W (these 5 pixels share the same parameter) while keeping the the other 70 pixels' I and W values same (I_C, W_C) and chosen so as to force the output in the specific range for a particular bucket of choice.

With these 6 curvefit functions, our approach to monitor the circuit output with the total of 150 different parameters of I and W $(I_0 - I_{74}, W_0 - W_{74})$ has three steps:

1. Select 1 f_{buc_s} out of the 5 bucket $f_{buc_1} - f_{buc_5}$ from the result $V_{OUT_{est}}$ from step 1, each bucket function f_{buc_i} covers the range $[\frac{i-1}{5}, \frac{i}{5}], (i \in [1,5])$.

2. Use the selected bucket curvefit function to calculate the final predicted convolution output. Here the bucket curvefit function adjusts the initial estimate to obtain more accurate estimate for behavior of each pixel:

$$V_{OUT_{pd}} = \sum_{i=0}^{74} \frac{f_{buc_s}(I_i, W_i) - f_{avg}(I_{C_s}, W_{C_s})}{5} + f_{avg}(I_{C_s}, W_{C_s})$$

This approach shows much more reliable output value as shown in the error rate bar plot Fig. 4.8(b), the error rate is below 3%.

To simplify the method, we combine step 1 and 2 into a single analytical equation that can be easily incorporated into ML frameworks like PyTorch. Selecting one out of five bucket curvefit functions effectively can be performed using a set of step functions. A significant drawback of using a step function is its non-differentiability at certain points, which poses a challenge for machine learning algorithms that require continuous derivatives for back propagation algorithm during training. To address this issue, we use sigmoid function $(\sigma(x))$ as an alternative to step function. The sigmoid function is advantageous because it is differentiable at all points, ensuring smoother transitions between output ranges. By combining sigmoid functions, as shown in Fig. 4.6(b), the bucket selection process based

$V_{OUT_{est}}$ can be accommodated into a single analytical equation:

$$V_{OUT_{pd\sigma}} = \sum_{i=1}^{5} \Bigg(\Big(\sigma(100(x - \frac{i-1}{5})) + \sigma(100(\frac{i}{5} - x)) - 1 \Big) \times \Big((\sum_{j=0}^{74} \frac{f_{buc_i}(I_j, W_j) - f_{avg}(I_{C_i}, W_{C_i})}{5}) + f_{avg}(I_{C_i}, W_{C_i}) \Big) \Bigg)$$

4.5 Results and Discussion

4.5.1 Analysis on Power, Latency and Bandwidth Reduction

Fig. 4.7 (a)-(c) and (d)-(f) show the simulation results for the analog convolution output for a single FPCA pixel and a group of 75 pixels, respectively, using TSMC 28nm HPC+ technology. The figures also show scatter plot comparing the linearity of the circuit output with that of ideal mathematical convolution output. As seen in Fig. 4.7 (c) and (f), the simulated convolution output of the FPCA pixel circuit exhibits fairly linear behavior. The small non-linearity associated with the analog convolution output needs to be accounted for in a machine learning framework compatible model such that the model can be used for algorithmic training of a deep learning network to mitigate any accuracy loss. Table 4.1 provides a qualitative comparison between the proposed FPCA and several previous in-pixel computing architectures. Since many quantitative metrics such as frame rate and power efficiency depend strongly on parameters like array size, pixel current, and technology node, we emphasize functional and architectural features instead. The proposed FPCA supports convolution, batch normalization, and ReLU operations within the pixel array and allows multi-channel processing in the first layer. Compared to previous analog and mixed-signal designs [12, 75], FPCA achieves higher functional completeness and enhanced reconfigurability while maintaining analog convolution capability. Moreover, by integrating computation and memory at the pixel level with a compact 4T cell design, FPCA offers improved flexibility for different network configurations and potential reductions in per-pixel energy consumption.

Work	SCAMP-5 [14]	PipSim [31]	Reconfig [15]	Mixed-Signal [12]	Senputing [96]	APS-P^{2}M [75]	FPCA (ours)
Supported Functions	MLP	Conv, ReLU	Conv	Binary Conv	Conv	Conv, BN, ReLU	Conv, BN, ReLU
# of Unit Cell Gates	100+T	4T	3T1C	4T	6T	(4+N)T	4T
Tech Node	180nm	45nm	180nm	180nm	180nm	22nm	22nm
Weight Form	Digital	Digital	Digital	Analog	Digital	Analog	Analog
Reconfigurability	High	High	High	Low	High	Low	High
Analog Conv	No	No	No	Yes	No	Yes	Yes
BN	No	No	No	No	No	Yes	Yes
ReLU	No	No	No	No	No	Yes	Yes

Table 4.1: Comparison of FPCA with a pixel array size of 1000 $\times$ 1000, kernel size = 5, stride size = 5, output channel size of 8 with related in-pixel, near-pixel computing works.

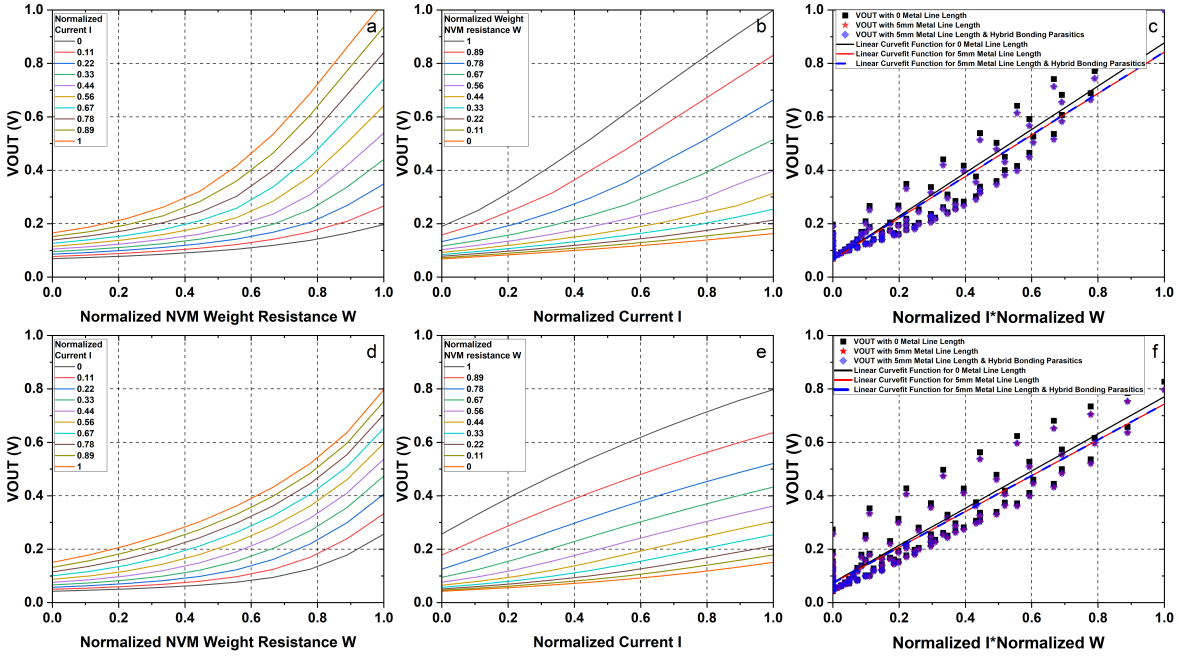


Figure 4.7: Simulation results of the FPCA circuit. (a) and (b) show the single-pixel analog output versus normalized NVM weight resistance W and input current I (representing light intensity). In (a), each curve corresponds to a different resistance , and in (b) each curve corresponds to a specific input current. Scatter plot (c) shows the linear curve-fit of the single-pixel data, where the black and red points represent different metal-line resistances between the shared-weight block and the pixel array, while the blue points incorporate both metal-line resistance and hybrid-bonding parasitics. Plots (d), (e), and (f) correspond to (a), (b), and (c) respectively, but for 75 simultaneously activated pixels (kernel size $5 \times 5 \times 3$) performing a full convolution operation.

Towards that end, the effectiveness of the novel curvefit bucket selection function approach in predicting circuit output is demonstrated in Fig. 4.8(b). This figure illustrates the error rate of estimated output voltage when applying the proposed curvefit method. The method involves simulation of convolution operation for 75 connected pixels with 150 randomly selected parameters (Ws, Is) spanning the entire range of weight and input current parameters. The comparison with simulation data shows that this method achieves accurate prediction of the analog output voltage, with an error rate of less than 3%. In addition to the architectural benefits, the electrical impact of Cu-Cu hybrid bonding was explicitly included in the FPCA circuit simulations. Based on parasitic values reported on hybrid-bonding for 3D integration [97], each vertical connection was modeled with a parasitic capacitance of approximately $0.17\mathrm{fF}$ and a resistance of approximately $50\mathrm{m}\Omega$ at a $4\mu\mathrm{m}$ pitch. As shown in Fig. 4.7(c) and (f), incorporating these parasitics produces negligible deviation in the analog convolution output compared to the case without hybrid bonding, confirming that the required 3D integration does not materially degrade FPCA accuracy for the array dimensions considered in this work.

4.5.1.1 Energy Analysis

The total frontend energy consumption $E_{FRONTEND}$ is quantified using Eq. 4.2, where $e_{PX} = 148\mathrm{pJ}$(calculated from simulation result), $e_{ADC} = 41.9\mathrm{pJ}$ [78], and E_{IO} represent the energy consumed per convolution operation, the energy per ADC read operation, and the total communication (IO) energy, respectively. The IO energy is further detailed in Eq. 4.3, where $e_{IO} = 12.34\mathrm{pJ/bit}$ indicates the energy cost per bit for the employed IO technology, specifically LVDS in this instance [98], $b_{ADC} = 8$ represents the ADC bit precision, h_o, w_o, and c_o specify the height, width, and number of output channels of the output activation map, respectively. Fig. 4.9(a) presents the normalized energy consumption for various stride sizes and number of channels in the output feature map for a constant kernel size of 5×5. The graph shows that employing strides of size 5 (non-overlapping) leads to maximum energy savings. The energy savings decrease as the stride size decreases, since lower stride size implies more number of convolution operation. Further, smaller number of output channels lead to higher energy savings. For reference, baseline energy number for a RGB camera without FPCA computing is shown in figure using a red dotted line. Thus, energy savings is achieved through use of higher stride size and lower number of channels [54, 79].

In contrast, increasing the output channel count to 32 does not lead to energy savings. This lack of improvement is due to an increase in the number of convolutional operations (N_C as shown in Eq. 4.1). These findings highlight that for FPCA design and in general for analog convolution in pixel, algorithm optimizations to reduce number of channels and increase number of strides is necessary along with hardware design. Thus, algorithm-hardware co-design is an imperative aspect of analog computing in pixel as well as FPCA design.

$$E_{FRONTEND} = N_C \times (e_{PX} + e_{ADC}) + E_{IO} \tag{4.2}$$

$$E_{IO} = h_o \times w_o \times c_o \times b_{ADC} \times e_{IO} \tag{4.3}$$

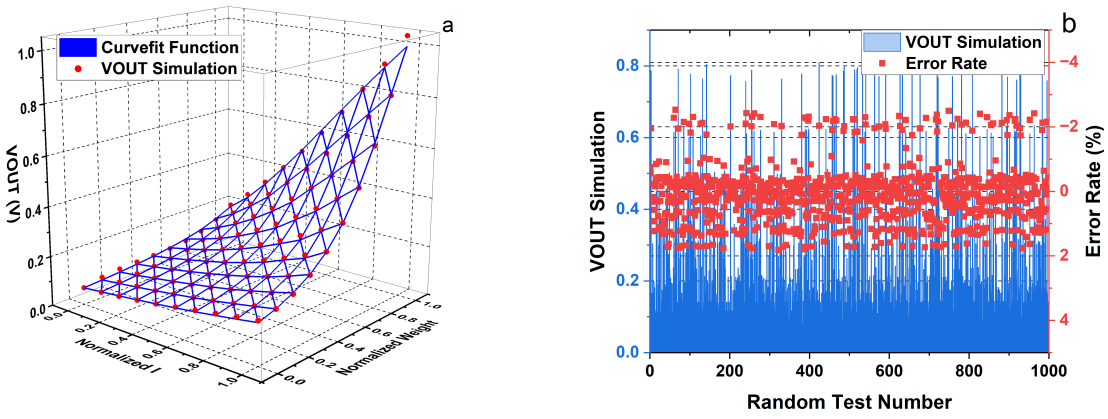


Figure 4.8: 3D curvefit function plot and the error rate bar plot. Plot (a) is the 3D curvefit function used to model the analog output behavior of the circuit for algorithmic use, plot (b) shows the error rate after applying the bucket select curvefit function with respect to the simulation data using random inputs (input current and weights) to each of the 75 pixels in TSMC 28nm HPC+ technology.

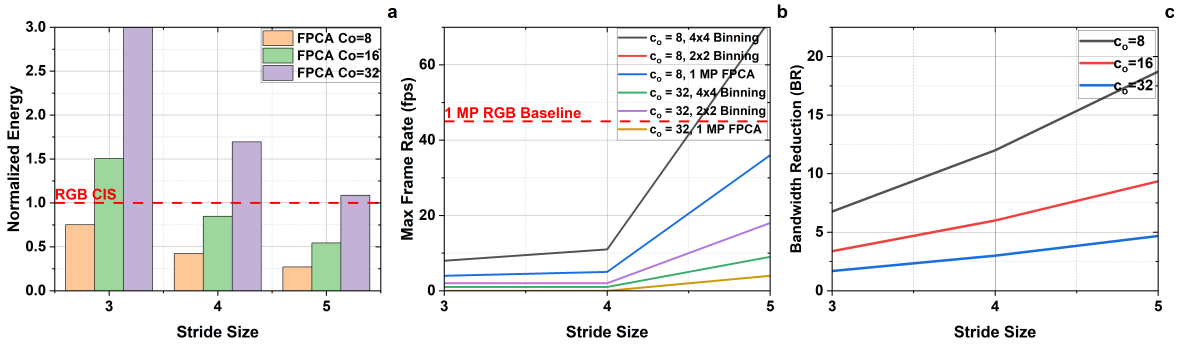


Figure 4.9: (a) Energy analysis showing normalized energy versus stride size for different numbers of output channels considering kernel size $n \times n = 5 \times 5$, (b) Latency analysis: Maximum frame rate versus stride size with different numbers of output channels and pixel binning considering the same kernel size of 5×5, and (c) is bandwidth reduction (BR) analysis where BR versus stride size for different numbers of output channels is plotted.

4.5.1.2 Latency Analysis

The latency $T_{FRONTEND}$ of FPCA convolution operation, depends on the number of read cycles as dictated by Eq.4.1. This latency, is further quantified using Equation 4.4, where T_{EXP} represents the exposure time, T_{ADC} the ADC read time, and T_{IO} the communication delay associated with IO pins. The IO delay, T_{IO}, is influenced by the ADC bit precision (b_{ADC}), the width of output activation map w_o, IO bandwidth ($BW_{IO} = 1Gbps$ [98]), and the total number of IO pads ($n_{IO(PAD)} = 24$) utilized on the chip as in Eq. 4.5. Fig. 4.9(b) shows the maximum frame rate ($\frac{1}{T_{FRONTEND}}$) achievable with an FPCA-enabled CIS, varying by stride number across different output channels and pixel binning scenarios, assuming a kernel size of 5. The plot shows that the maximum frontend frame rate of the FPCA model is generally lower than that of conventional RGB CIS. This is due to the inclusion of both positive and negative weights in the first convolutional layer and the fixed max kernel size for shared weight block. Nevertheless, by optimizing the stride, reducing output channels, and implementing pixel-level binning, higher frame rates are attainable (as demonstrated with $c_o = 8$ and 4×4 binning for stride number = 5).

$$T_{FRONTEND} = N_C \times (T_{EXP} + T_{ADC} + T_{IO}) \tag{4.4}$$

$$T_{IO} = \frac{w_o \times b_{ADC}}{BW_{IO} \times n_{IO(PAD)}} \tag{4.5}$$

It is important to note that for in-pixel computing, as defined in [78] the frontend latency is considered as the total computation time for all activations of the output feature map within the first convolutional layer. To reduce the latency compared to conventional CIS significant reductions need to be made in the output feature map dimensions, such as through fewer channels, larger strides, or binning. [75, 76] Moreover, reduction in exposure time can also increase the frame rate, irrespective of the specific FPCA configuration employed.

4.5.1.3 Bandwidth Reduction Analysis

For Bandwidth Reduction in FPCA, the data bandwidth reduction (BR) can be estimated using Equation 4.6 [78]. Here, I and O represent the number of elements in the input RGB image and the output activation map of the first convolutional layer, respectively. The input I is calculated as $h_i \times w_i \times 3$, where h_i and w_i are the height and width of the input image, and the factor 3 is for the three RGB channels. The output O is derived from Equation 4.7 and Equation 4.8, where h_o, w_o, and p denote the height and width of the output activation map and the padding, respectively. The factor $\frac{4}{3}$ in Equation 4.6 accounts for the compression efficiency from converting the Bayer RGGB pattern, commonly used in image sensors, to the standard RGB format. And the term $\frac{12}{b_{ADC}}$ reflects the conversion from a higher bit depth, typically 12 bits per color channel in raw sensor outputs, to the bit precision ($b_{ADC} = 8$) used by the peripheral ADC, which is set to 8 bit activations for deep learning applications.

$$BR = \left(\frac{I}{O}\right)\left(\frac{4}{3}\right)\left(\frac{12}{b_{ADC}}\right) \tag{4.6}$$

$$O = h_o \times w_o \times c_o \tag{4.7}$$

$$h_o(w_o) = \frac{h_i(w_i) - n + 2 \times p}{S} + 1 \tag{4.8}$$

Fig. 4.9(c) illustrates the estimated data bandwidth reduction (BR) versus stride size for a kernel size of 5×5 with various numbers of output channels. The graph shows that the FPCA approach can provide a significant reduction in data bandwidth compared to conventional CIS, particularly when employing large strides and fewer output channels. Additionally, improved BR values can be achieved by implementing pooling (either max or average) at the periphery following the output from the first convolution layer in the FPCA design.

4.5.2 3D Integration

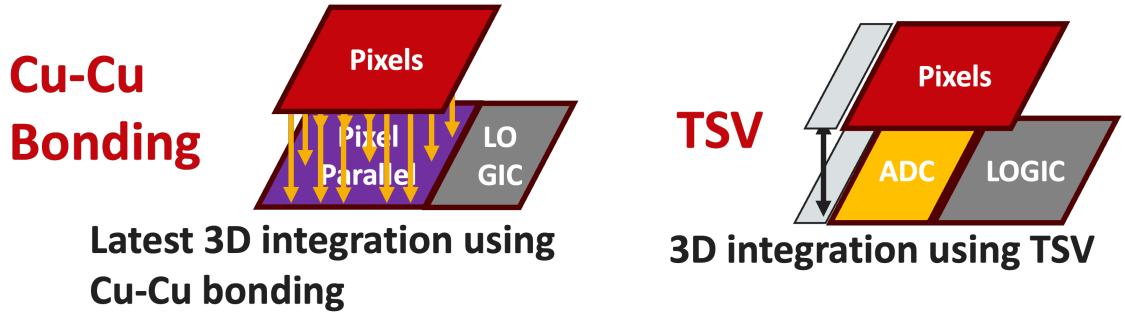


Figure 4.10: 3D integration techniques where left hand-side is cu-cu bonding and right hand-side is TSV.

3D integration is a critical enabler for the FPCA circuit, allowing in-pixel computing without compromising pixel density. In current camera sensors, 3D integration has historically been utilized for various photography applications, but its adoption in in-pixel computing marks a significant shift toward embedding intelligence within camera pixels. Traditional implementations using TSVs as in Fig.4.10, such as those described in [43], have demonstrated a reduction in chip area compared to 2D architectures. However, TSVs are constrained by their placement outside the pixel arrays, resulting in underutilized areas. Recent advancements in fine-pitch Cu-Cu bonding have addressed these limitations by enabling pixel-parallel architectures. Unlike TSV-based configurations, where signal lines are routed to a peripheral logic layer, Cu-Cu connections as shown in Fig.4.10 are integrated directly, increasing interconnect density and enhancing the adaptability of the system. Samsung's design [99] showcases this approach, utilizing two small-pitch Cu-Cu interconnections per pixel to connect the pixel-wise ADCs and in-pixel digital memory, achieving a pixel pitch of less than 5μm. Similarly, Sony has demonstrated a 3μm pitch with Cu-Cu bonding in their image sensors [100], proving the electrical reliability and viability of ultra-fine-pitch Cu-Cu connections in commercial applications. By employing Cu-Cu bonding for the FPCA circuit, weight NVMs can be implemented on a separate die, which is heterogeneously integrated with the back-side-illuminated (BI) CMOS Image Sensor (CIS) die. This heterogeneous integration uses techniques such as die-to-die or die-to-wafer bonding, allowing the weight NVM die to be fabricated using advanced planar or non-planar technology nodes, maximizing storage capacity for larger weight kernels and channel sizes. This approach ensures

high pixel density while supporting the FPCA circuit's computational demands, enabling more intelligent and efficient imaging systems.

4.6 Future Work

4.6.1 Algorithm Testing

Future work on the FPCA framework will delve into the intricate balance between hardware reconfigurability and algorithmic precision, with a specific focus on accuracy analysis at the algorithm level. The goal is to understand how FPCA-implemented systems can preserve accuracy through adaptive training strategies that account for hardware-induced non-linearities. This aspect becomes particularly critical in complex applications requiring granular feature extraction, such as multi-object tracking from the BDD100K dataset, where larger dimensionality reductions can adversely impact performance. Note that although the present work does not include new algorithm-level accuracy evaluations, strong prior evidence supporting the expected performance of FPCA comes from our earlier P2M-based architectures [75, 76, 78, 80] which employ the same analog-first, digital-rest computation pipeline. In those systems, only the first convolution layer is computed in analog while all subsequent layers operate in the high-precision digital domain. End-to-end training was shown to successfully compensate for analog non-idealities—including limited precision and nonlinearity—achieving accuracy comparable to digital baselines on challenging datasets such as BDD100K [76], for large input resolutions and multi-object detection/tracking tasks. Because FPCA generalizes the same computation principle while providing a more scalable and reconfigurable kernel architecture, similar or improved accuracy is expected once the FPCA bucket-select model is integrated into an end-to-end training flow, which is a key direction of future work.

4.6.2 Column-wise Multi-Channel Weight Array Self-Shift Operation

The FPCA architecture requires deploying a dedicated Weight Array for each column within the pixel array. These Weight Arrays are identical across columns, storing uniform weight values essential for conducting in-pixel convolution operations. To enhance operational

efficiency, particularly in reducing the latency associated with updating weight values, the introduction of a self-shift mechanism within each weight array warrants exploration. This approach aims to shorten the process latency of weight value adjustment, potentially minimizing the preparation time required for transitions between different convolutional tasks.

4.7 Conclusion

In conclusion, this paper has introduced a Field-Programmable Pixel Array (FPCA), a novel approach to in-sensor and in-pixel computing that pushes the potential of pixel arrays within CMOS image sensors to wide range of modern convolutional deep learning networks. Unlike traditional pixel arrays, FPCA provides dynamic adaptability in weight values, kernel sizes, channel sizes, and stride sizes, addressing the rigid constraints that have limited intelligent sensor design and functionality without sacrificing pixel area by implementing weight array on a separate die connected with TSV or Cu-Cu bonding. Moreover, the FPCA architecture integrates the capability of region skipping, allowing for selective processing that enhances efficiency and reduces unnecessary computations. The paper also presents a novel bucket select curvefit based analog modeling approach that can accurately capture the non-linear behavior of analog computing. The resultant model is machine learning framework compatible and can be used for training deep learning networks to mitigate accuracy loss. Advantageously, the developed model is applicable to analog computing in general beyond the presented FPCA use-case, including memristive crossbar arrays. Further, our analysis shows the dependence of energy, latency and bandwidth metrics on typical deep learning parameters, highlight the importance of algorithm-hardware co-design for extreme-edge intelligence applications.

In summary, the work provides a versatile framework with potential applications across wide range of various computer vision tasks, spanning from simple object detection to more complex scene understanding. Its flexibility and efficiency may contribute to the development of intelligent devices designed to perform sophisticated image processing tasks directly at the point of data generation.

Chapter 5

A Pathway to Near Tissue Computing through Processing-in-CTIA Pixels for Biomedical Applications (CTIA-IPC)

5.1 Introduction

Edge computing inside image sensors is emerging as a compelling solution to the bandwidth, power, and latency bottlenecks of off-chip data transfer, constraints that are especially critical in real-time biomedical settings such as minimally invasive surgery (MIS), where extremely low power budgets are essential. In these scenarios, transmitting raw high-resolution image data from implantable or catheter-based sensors to external processors imposes unsustainable energy and bandwidth costs. In-Pixel Computing (IPC) architectures aim to address this by embedding processing and memory directly into the pixel array to accelerate operations such as convolutions that are essential for machine learning applications and they provide bandwidth reduction through spatial stride and channel compression. However, most prior IPC systems suffer from limited computational precision, output non-linearity [75, 77], or degraded performance under low-light conditions [12, 14, 31]. This makes them unsuitable for dynamic surgical environments where robustness and accuracy are essential.

We present CTIA-IPC, a capacitive transimpedance amplifier (CTIA) based in-pixel computing architecture that enables CNN inference directly within the image sensor. Leveraging linearity, low noise, and high sensitivity [5, 101, 102] of CTIA pixels, our design integrates

weight-to-time conversion, 3D-stacked SRAM, and modifications to peripheral ADCs to implement efficient multiply-accumulate (MAC) operations inside the pixel array. The CTIA-IPC is built on IPC's architecture that provides bandwidth reduction and it also supports multi-channel, multi-bit convolution, batch normalization (BN), and Rectified Linear Unit (ReLU) activation in a fully on-chip pipeline. We validate its utility in a real-world context by applying it to surgical instrument segmentation using the EndoVis dataset, employing a co-designed vision model trained with analog-aware quantization.

Our key contributions include i) a dual-mode CTIA-IPC pixel array supporting both imaging and convolution, ii) a weight-to-time converter mapping signed weights to individual pixel exposure time, thereby enabling in-pixel MAC operations iii) column-level integration of BN and ReLU within a single-slope ADC (SS-ADC), iv) 3D-stacked SRAM (TSV/Cu–Cu hybrid bonding) over CTIA pixels for dense, kernel (weight) storage, and v) an algorithm-hardware co-design pipeline that reduces off-chip bandwidth by 12× and achieves 1.98 GOPS at 3.39 GOPS/W with only a 1.3–2.5% IoU drop on the EndoVis [103] MIS benchmark. The remainder of this paper presents the architecture (Sec. 5.2), dataset and methodology (Sec. 5.3), experimental results (Sec. 5.4), and conclusions.

5.2 Proposed CTIA Pixel Architecture for In-Pixel Computing

5.2.1 Proposed CTIA-IPC Design

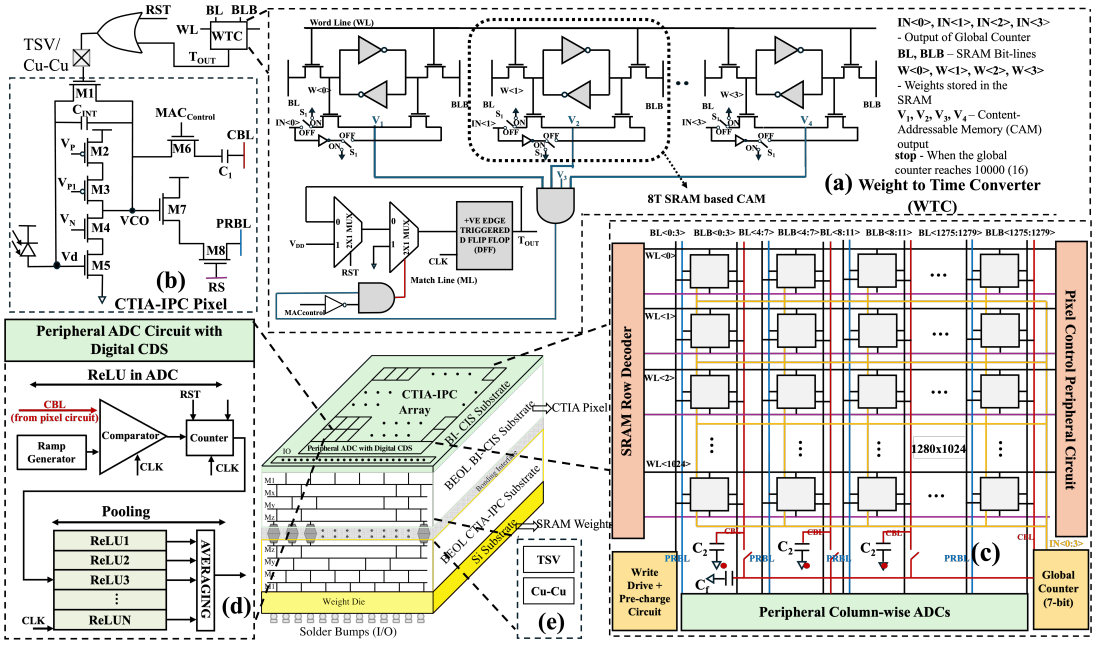


Figure 5.1: CTIA-IPC architecture where (a) 4-bit Weight to time converter (WTC); (b) CTIA-IPC pixel unit circuit diagram for both multiplication and regular readout; (c) 1280×1024 CTIA-IPC Array with peripheral readout circuits; (d) Block diagram of Single Slope Analog to Digital Converter (SSADC) with Digital CDS architecture for ReLU and Pooling functionalities; (e) is the connection between the two dies using either Through-Silicon Vias (TSV) or Copper-Copper bonding (Cu-Cu) for 3D integration.

5.2.1.1 CTIA-IPC Pixel Circuit Design

Fig. 5.1(b) illustrates the CTIA pixel circuit, where the photodiode is maintained at a constant voltage by the operational transconductance amplifier (OTA: M2-M4). The OTA amplifies the photodiode voltage during the integration phase. The bias voltages (V_P, V_{P1}, V_N) required by the OTA are supplied by globally shared bias generator circuits across the array. During the reset phase, the CTIA output node VCO is set to the photodiode voltage Vd, which is driven to a reset voltage V_{RST}. In this phase, the reset transistor M1 is turned on,

forcing zero voltage across the integration capacitor C_{int}. This leads to charge sharing between Vd and VCO, equalizing both nodes to the same potential. Once the reset phase ends, the Vd node is held constant, while the VCO node begins to discharge as photocurrent flows from the photodiode supply (VDD) through the integration capacitor. At the end of the weight-dependent integration period (triggered by the next RST pulse), the CTIA output node VCO holds a voltage proportional to the product $W \times I$. This per-pixel multiplication result is then transferred onto the column CBL, where contributions from all pixels in the kernel window are charge-accumulated to form the MAC output (detailed in Sec. 5.2.2). The timing of the reset switch M1 is jointly controlled by the global RST signal and the WTC output, ensuring that each pixel integrates for the correct weight-encoded duration before contributing to the CBL.

5.2.1.2 Weight-to-Time Converter (WTC)

The weight–to–time converter (WTC) (Fig. 5.1(a)) enables each CTIA pixel to implement multiplication by mapping its stored neural weight to an integration time. The photodiode current is integrated over this weight-dependent duration, so the accumulated charge at V_{CO} represents the product of photocurrent and weight.

Circuit Description: Each pixel stores a 4-bit weight in an SRAM-based content-addressable memory (CAM) 3D integrated with each pixel. The CAM is built from 8T SRAM cells with a control switch S_1: when on, it behaves as a 6T SRAM for weight write and storage; when off, it enables compute mode, performing XNOR comparisons between stored bits and inputs (IN0–IN3) driven by a 7-bit global counter (Fig. 5.1(c)) shared across the array. Four consecutive global-counter bits (e.g., bits 0–3, 1–4, etc.) define the WTC resolution because each unique 4-bit pattern corresponds to one discrete integration window. Shifting the starting bit effectively changes the step size of this window, enabling weight-dependent integration durations. Regarding batch normalization (BN), the multiplicative BN scale factor is fused into the trained convolutional weights during the hardware–algorithm co-design stage. The additive BN offset is implemented in hardware by initializing the digital CDS counter to a non-zero reset value, which shifts the ADC output by the required constant.

Handling signed weights: To support positive and negative CNN weights [104], we extend the WTC using column-parallel single-slope ADCs (SS-ADCs) with digital correlated-double sampling (CDS, Fig. 5.1(d)). Positive weights are encoded during up-counts and negative

weights during down-counts, so each MAC requires two cycles (represented in Fig. 5.2's Digital CDS OUT panel). Negative results are clipped to zero, inherently implementing ReLU, while batch normalization is embedded by initializing the CDS counter with BN additive factors.

Area and integration considerations: The baseline CTIA pixel occupies approximately $100\,\mu m^2$ [5], at the 22 nm technology node, the additional circuitry required for CTIA-IPC introduces only a small fractional overhead. Specifically, the 4-bit 8T SRAM weight storage requires approximately $0.6\mu m^2$ in total (i.e., $0.15\,\mu m^2$ per bit [105]). The WTC control logic including flip-flops and combinational logic gates adds roughly $6\,\mu m^2$. The MOM CBL capacitor contributes an estimated $10.5\,\mu m^2$. Summed together, the total added area per CTIA-IPC unit is approximately $17–18\,\mu m^2$, which corresponds to less than 20% of the baseline CTIA pixel footprint. With 3D integration using TSV or Cu–Cu hybrid bonding, the SRAM array and WTC logic can be shifted to an upper tier, leaving only minimal routing components within the pixel layer. As a result, the effective active area in the pixel tier remains comparable to that of standard CTIA sensors, thereby preserving the photodiode fill factor. This demonstrates that CTIA-IPC achieves in-pixel MAC functionality with negligible impact on pixel density or imaging performance.

5.2.2 CTIA In-Pixel Compute Accelerator

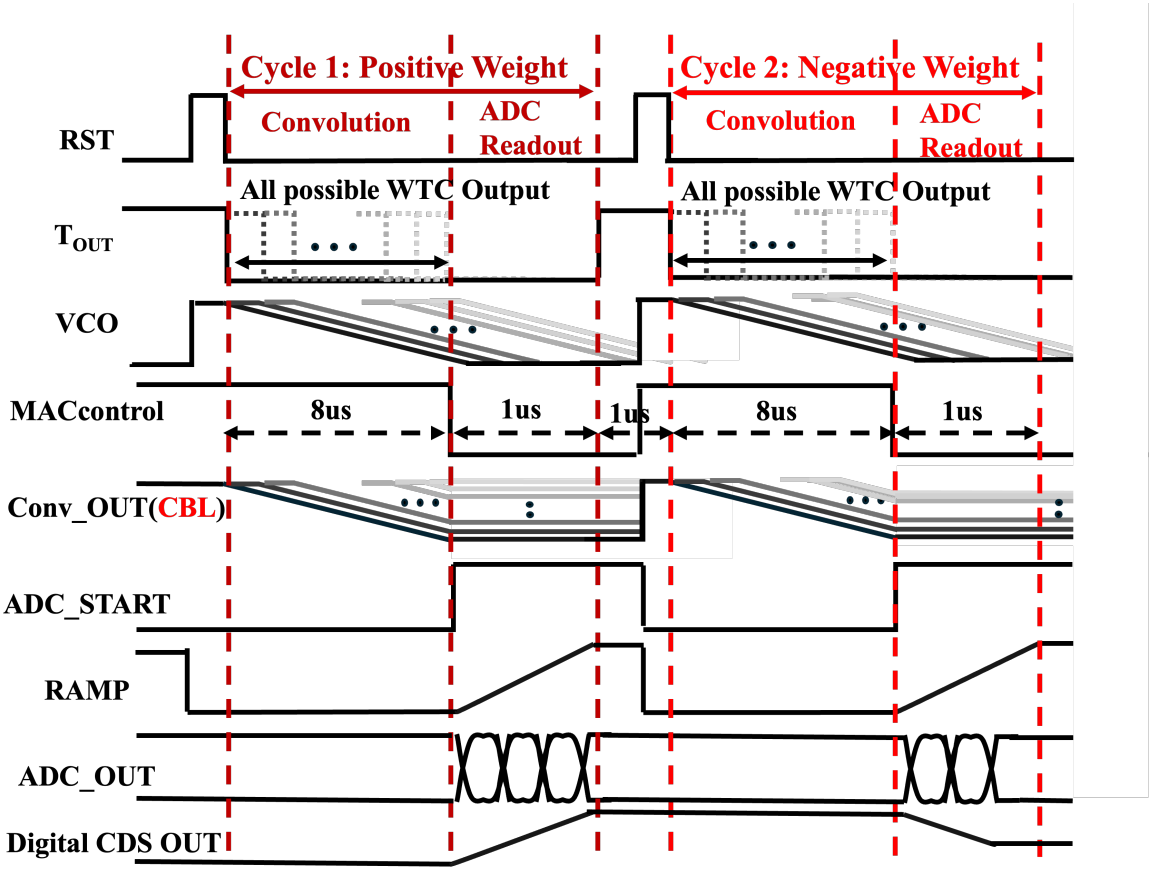


Figure 5.2: A typical timing diagram for CTIA-IPC's convolution computation showing double sampling for positive and negative weights. The diagram includes control signals from the Global Counter, CTIA-IPC unit, and SS-ADC: RST, $\mathrm{MAC}_{\mathrm{Control}}$, ADC_START, and the pixel/column outputs $\mathrm{T}_{\mathrm{OUT}}$, VCO, Conv_OUT, RAMP, and ADC_OUT. The *Digital CDS_OUT* panels illustrates the CDS output: its digital value first increases then decreases during the positive-first negative-second 2 cycle for convolution.

The proposed accelerator integrates a 1280×1024 CTIA-IPC pixel array (Fig. 5.1(c)), where each pixel combines a standard CTIA pixel with a WTC and a capacitive bitline (CBL) for charge accumulation (Fig.5.1(b)). This design allows the array to function either as a conventional image sensor, using source-follower readout (M7-M8 in (Fig. 5.1(b))), or as a massively parallel in-pixel convolution engine (M6, CBL).

Column-level MAC: During convolution, each pixel performs an analog multiplication be-

tween the photodiode current and its stored weight at node VCO, using the WTC described in Sec. 5.2.1 and contributes charge onto its column bus (CBL). The CBL voltages are then combined using a capacitive switching network that transfers the accumulated charge onto a metal–oxide–metal (MOM) capacitor. The aggregated result is digitized by a 4-bit SS-ADC. This capacitive accumulation follows the equation: $V_{\text{ADC\_IN}} = \frac{V_{VCO1}+V_{VCO2}+\cdots+V_{VCON}}{4+\frac{2C_2}{C_1}+\frac{C_F}{C_1}}$, where C_1 is the capacitor within the CTIA-IPC unit in Fig. 5.1(b), C_2 and C_f are column capacitors as shown in Fig. 5.1(c), V_{VCO1}, V_{VCO2}, to V_{VCON} are the multiplicative values generated by each CTIA-IPC unit on node VCO (Fig. 5.1(b)).

Array-level convolution: To determine how many pixels can be activated in parallel, we compute the maximum number of convolution windows that can be placed along a row without overlap. For an input row length i with kernel size k, padding p, and stride s, the number of spatially valid convolution windows per row is $\frac{i-k+2p}{s\times\text{lcm}(k,s)}$ (as kernel and stride boundaries do not always align), this determines how many windows can be evaluated simultaneously without interference. Each window requires k^2 pixels, and the RGGB color filter array contributes a factor of 4. Thus, in one compute cycle, the array activates $\frac{(i-k+2p)}{s\times\text{lcm}(k,s)} \times k^2 \times 4$ pixels in parallel, whose outputs are accumulated along the corresponding CBLs and forwarded to the column ADC.

System-level integration: With this pipeline, convolution, BN, and ReLU activation are all performed in and near pixel. An optional on-chip pooling layer can further reduce spatial resolution. Together, these operations allow the CTIA-IPC array to map CNN layers directly on the sensor with minimal area and energy overhead, meeting real-time demands for medical tasks such as anomaly detection and lesion segmentation.

5.3 Medical Imaging Application

5.3.1 Details of the Dataset & Tasks

To evaluate the practical utility of our CTIA-IPC architecture for real-time biomedical imaging, we benchmark it on a representative task: surgical instrument segmentation in robot-assisted minimally invasive surgery (MIS) [106]. This task involves high-resolution and low-light endoscopic imaging conditions that align well with the CTIA-IPC's strengths in linearity, sensitivity, and in-pixel computation. We evaluate on the EndoVis benchmark [103], which presents real-world challenges (occlusions, specular highlights, and uneven

illumination) on high-resolution (1280×1024) endoscopic video. Our training set includes eight 225-frame sequences, and evaluation uses two separate 300-frame sequences, all with pixel-wise instrument masks. The CTIA-IPC's linear analog MAC operations and integrated BN/ReLU support stable feature extraction under such lighting variability, preserving spatial detail critical for accurate segmentation and enabling efficient inference at the sensor edge.

5.3.2 Algorithm-Hardware Co-Design

To interface our analog compute model with a practical vision pipeline, we adopt a hardware-algorithm co-design approach grounded in SPECTRE-level simulation. The CTIA-IPC circuit, described in Section II, implements analog convolution that despite showing high linearity deviates from ideal linear behavior due to device and circuit limitations. Unlike prior works that overlook device non-idealities [26, 27], we simulated the CTIA-IPC circuit using 22nm GlobalFoundries technology, sweeping parameters such as transistor width and photodiode current. We integrate this behavior into the network by replacing TernausNet [107]'s original VGG11-based first convolution layer (3×3, stride 1, 64 channels) with a CTIA-modeled analog MAC layer (7×7, stride 2, 16 channels). The analog MAC is implemented through a curve-fit surrogate function following [75, 76]. The analog output is then processed by batch normalization and a quantized ReLU, where the activation is the 4-bit SS-ADC output, minimizing pixel-level bandwidth. Moreover, we incorporate three noise sources to model realistic sensing-to-ADC behavior: (1) 1% full-scale Gaussian noise added to the input image to simulate temporal and fixed-pattern noise; (2) 2.9% Gaussian noise applied to the CTIA curve-fit outputs, matching measured CTIA variability; (3) 1.5% noise injected into the 4-bit quantization range variation to approximate ADC noise. During inference, batch normalization (BN) is fused into the preceding convolution and succeeding activation layers. Consider a BN layer with trainable parameters γ, β, and running mean and variance μ, σ. This BN layer implements a linear function during inference:$Y = \gamma\frac{X-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta = \left(\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}\right)X + \left(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}\right)$. We fuse the scale term $A = \frac{\gamma}{\sqrt{\sigma^2+\epsilon}}$ into the convolutional layer weights, adjusting the pixel weight tensor to $A \cdot \theta$, where θ is the trained weight tensor. Additionally, the analog comparator threshold is adjusted by $B = \left(\beta - \frac{\gamma\mu}{\sqrt{\sigma^2+\epsilon}}\right)$, such that the ReLU threshold becomes $-B$. We train on EndoVis using Adam [108] with an initial learning rate of $1e{-}4$, decayed to $1e{-}5$ after 10 epochs. The network is trained for 20 epochs on center-cropped images. For binary segmentation, all tool classes are merged into

a foreground mask and optimized by Jaccard loss. For parts and instrument segmentation, an additional cross-entropy loss is applied.

As the in-pixel SRAM used for weight storage is volatile, trained weights are preloaded into the array at initialization via a standard write interface. The architecture is designed for inference-only operation, with no on-chip learning or weight updates. During inference, the SRAM operates in CAM mode to support compute cycles using the stored weights.

5.4 Results and Discussion

System Overview	Technical Details		Performance Metrics		
Works	Supported Func.	Tech Node	Power/Pixel(W)	Efficiency (OPS/W)	Linearity (R^2)
APS-P^2M [75]	Conv, BN, ReLU	22nm	0.18µ	0.4T	0.863
Reconfig [15]	Conv	180nm	0.025µ-0.11µ	1.41-3.37T	-
DROIC [101]	Low Light	180nm	0.71µ	-	-
CTIA-Fluo [102]	Low Light	500nm	0.145µ	-	-
CTIA-CDS [5]	Low Light	350nm	3.85µ	-	-
CTIA-IPC (ours)	Low Light, Conv, BN, ReLU	22nm	3.26µ	3.39G	0.995

Table 5.1: Comparison of CTIA-IPC with related CTIA and In-Pixel Computing works.

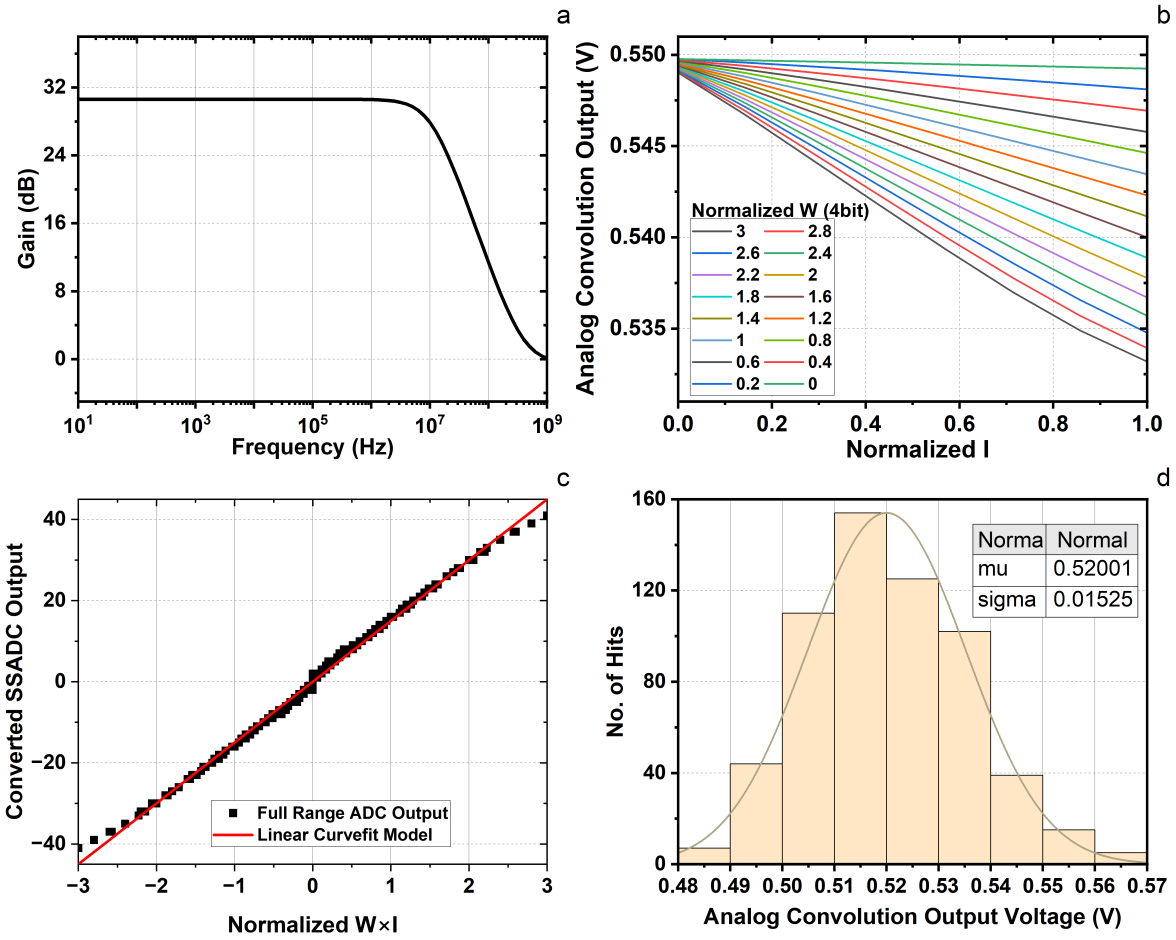


Figure 5.3: CTIA-IPC circuit simulation results: (a) OTA gain; (b) analog MAC output vs. normalized weight W and input current I (representing light intensity) where each line corresponds to different digital weight values that are translated from the WTC, (c) SS-ADC linearity after digital CDS; (d) Monte-Carlo variation analysis of the MAC output.

5.4.0.1 Circuit Simulation Results

SPECTRE simulations in 22 nm FDSOI verify that the proposed CTIA-IPC pixel performs linear analog MAC operations under low-light conditions, demonstrating high linearity (as shown in Table 5.1's Linearity column). The OTA is implemented in 22 nm FD-SOI where $g_m r_o$ limits open-loop gain to 25–40 dB, achieves 30 dB (Fig. 5.3(a)), which is near the upper bound permitted by the 0.8 V supply; device lengths, bias currents, and cascoding were tuned to maximize gain while keeping all devices in saturation. The resulting CTIA gain error remains small, supporting accurate in-pixel MAC. Fig. 5.3(b) shows that the analog convolution output maintains strong linearity with respect to $W \times I$ across multiple weights

and photocurrents, and the SS-ADC output in Fig. 5.3(c) preserves this linear behavior through digitization via digital CDS. Fig. 5.3(d) is a 1000-point Monte Carlo analysis on convolution output for the analog IPC array. For a representative 3×3 kernel MAC operation, the nominal analog output voltage is 0.51 V. Under process-induced variations, the output voltage spans from 0.48 V to 0.57 V, with a standard deviation (σ) of 0.0152 V. This narrow distribution indicates that the impact of process variations on the IPC output is minimal, thereby validating the robustness of the proposed IPC array under realistic fabrication conditions.

5.4.0.2 Performance Metrics

For biomedical computation Bandwidth Reduction (BR) ($= \frac{I}{O}\frac{4}{3}\frac{12}{N_b}$)is the key metric as BR can directly affect the efficiency of the pixel circuit, Our CTIA-IPC array achieves a BR of 12.08 over raw pixel output and it exhibits a 3.26 μW power consumption per pixel (For a k=7 kernel size with p=0, s=2, c_o=16, N_b=4, p_s=2, $I = 1280 \times 1024$, $O = (\frac{i-k+2p}{s}+1)^2/p_s^2 \times c_o$). The compute signal output was sampled from the output of a 4-bit SS-ADC. From simulation, including pixel readout and digital CDS energy, this work achieves a throughput of 1.98 GOPS with an energy efficiency of 3.39 GOPS/W. Using these performance metrics, we compare our design to prior IPC and CTIA-based architectures, as summarized in Table 5.1. Although the reported energy efficiency appears lower compared to existing 3T and 4T-based IPC architectures [15, 75], this result primarily arises from our targeted biomedical near-tissue application which requires CTIA pixels that are inherently more complex compared to 3T/4T pixels. Our CTIA pixels based IPC approach is well suited for biomedical imaging conditions, as compared to simpler pixel designs; thus, direct efficiency comparisons to earlier IPC works are less applicable. As for power consumption, prior CTIA pixel designs optimized purely for low-light imaging [101, 102] report lower power/pixel than CTIA-IPC. However, the primary advantage of CTIA-IPC over prior CTIA sensors lies in its ability to perform accurate in-pixel computing while preserving the intrinsic linearity and low-light robustness of CTIA pixels. Consequently, comparing all architectures on the same EndoVis images, captured independently of sensor physics, cannot fully reflect architectural differences. A fairer evaluation requires datasets with multiple resolutions and, ideally, paired recordings from both standard imagers and CTIA-based sensors; Such data would enable studying how pixel-level characteristics translate into segmentation

accuracy. Our CTIA-IPC design establishes the feasibility of CTIA-based in-pixel computing for surgical imaging, but additional datasets are needed to study accuracy dependence on pixel characteristics.

Task	Binary Seg.			Parts Seg.			Instrument Seg.		
Model	IoU(%)	Dice(%)	SEN(%)	IoU(%)	Dice(%)	SEN(%)	IoU(%)	Dice(%)	SEN(%)
TernausNet [107]	80.14	88.07	95.48	77.18	86.64	86.15	30.10	42.34	84.73
APS-P^{2}M 3× [75]	74.36	85.30	91.10	73.11	83.69	82.73	27.62	41.01	82.02
APS-P^{2}M 12× [75]	74.61	85.46	92.15	72.46	83.39	86.04	24.12	36.87	76.23
CTIA-IPC 3× (ours)	78.76	88.24	94.58	75.37	86.07	86.05	29.69	42.10	84.61
CTIA-IPC 12× (ours)	77.99	87.64	95.30	75.93	85.65	83.93	28.60	41.52	82.20

Table 5.2: Performance of the baseline models and the proposed CTIA-IPC on EndoVis segmentation benchmarks.

5.4.0.3 Task Accuracy

We evaluated TernausNet [107], APS-P^{2}M [75], and the proposed CTIA-IPC under 3× and 12× bandwidth reduction, where the 3× model removes the first-layer max pooling. APS-P^{2}M is processed similarly by replacing the first convolutional with an analog MAC surrogate. Results, including Intersection over Union (IoU), Dice coefficient, and Sensitivity (SEN) in percentages, are summarized in Table 5.2. CTIA-IPC consistently outperforms APS-P^{2}M across all tasks. In binary and parts segmentation, IoU improves substantially (e.g., 78.76% vs. 74.36% under 3×). APS-P^{2}M suffers at 12×, while CTIA-IPC maintains stable accuracy. For instrument segmentation, CTIA-IPC achieves higher IoU and Dice at both compression levels. These results show the nonlinearity in IPC can degrade MAC accuracy and downstream CNN performance. In conventional APS- or source-follower–based IPC architectures, the introduced nonlinear distortions in the analog MAC operation alter the effective scaling of feature maps, leading to reduced segmentation accuracy, especially in low-light conditions where analog signals are weaker. In contrast, CTIA pixels maintain with minimal nonlinearity, preserving the proportionality between input weights and analog outputs, ensuring high MAC fidelity and stable performance across all segmentation tasks. When compared with the baseline model, the use of 7×7 kernel introduces controlled

spatial aliasing, enhancing edge preservation under low-light conditions, with minor impact on fine-detail precision. Our CTIA-IPC can limit IoU degradation to only 1.3%–2.5%.

5.5 Conclusion

We introduced CTIA-IPC, the first in-pixel CTIA accelerator capable of performing multi-bit, multi-channel convolutions with integrated batch normalization and 4-bit ReLU. A 3D-stacked SRAM layer stores weights directly above each pixel, preserving array density while enabling dense in-pixel computation. With linearity-aware 4-bit training, CTIA-IPC achieves segmentation accuracy on the EndoVis dataset within 1.3–2.5% of the software baseline, while reducing sensor-to-processor bandwidth by 12×. Our approach demonstrates a throughput of 1.98 GOPS at an energy efficiency of 3.39 GOPS/W, providing a high-linearity, low-light computing solution tailored for real-time surgical and diagnostic imaging.

Chapter 6

P^{2}M Silicon Prototype

The preceding chapters developed the P^{2}M paradigm through circuit simulation and algorithm–circuit co-design on the GlobalFoundries 22nm FD-SOI technology node. To validate the feasibility of the proposed architecture beyond simulation, we taped out a P^{2}M test chip, the first silicon prototype from the Compute Lab at UW–Madison. This chapter describes the physical implementation, the test infrastructure built around the fabricated chip, and the current status of silicon characterization.

6.1 Chip Design and Physical Implementation

The P^{2}M test chip was designed and taped out in the GlobalFoundries 22nm FD-SOI process. Translating the circuit architecture described in Chapter 3 from schematic-level SPICE simulations to a fabrication-ready layout required establishing a complete digital and mixed-signal design flow in the lab, from register-transfer level (RTL) specification through synthesis, place-and-route, and final GDSII generation. This included integrating the analog pixel array with digital control logic for row/column selection, channel multiplexing, and SS-ADC timing generation.

Fig. 6.1 shows the final chip layout. The design integrates the weight-embedded pixel array with column-parallel SS-ADC readout circuitry and the digital control blocks needed to orchestrate the multi-phase P^{2}M convolution operation described in Section 3.3. Peripheral I/O pads provide interfaces for power supply, clock, configuration signals, and digital readout of the convolution outputs.

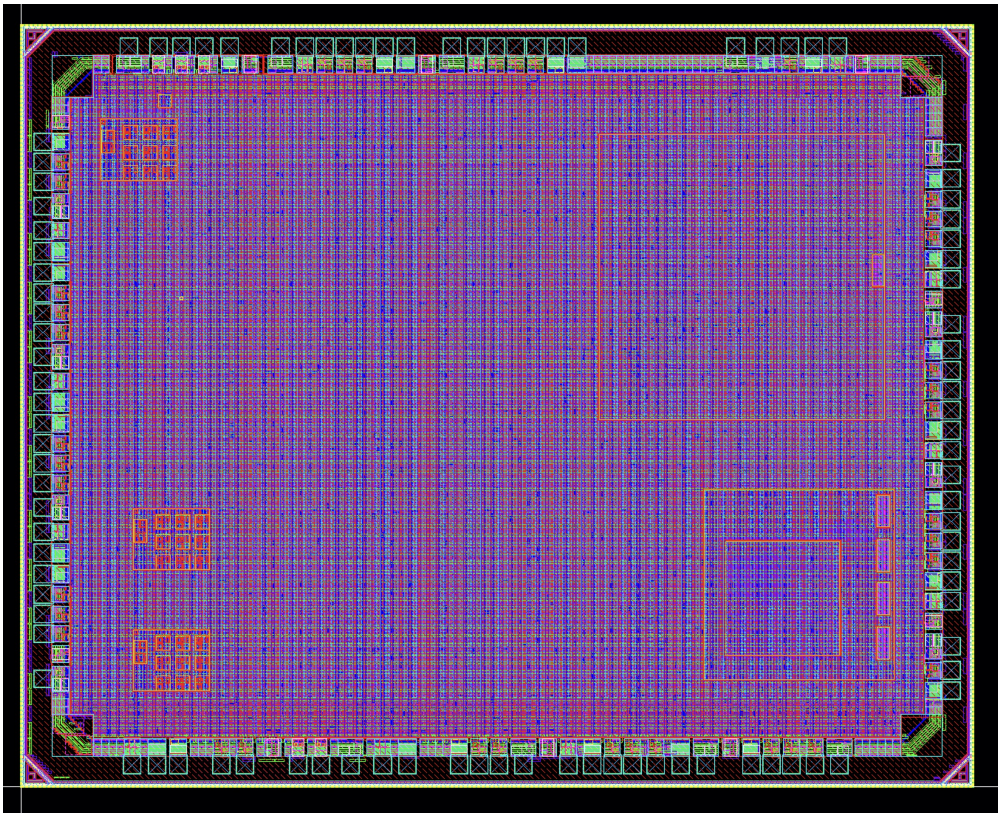


Figure 6.1: Final layout of the P^{2}M test chip in GlobalFoundries 22nm FD-SOI technology. The design integrates the weight-embedded pixel array with column-parallel SS-ADC readout and digital control logic.

6.2 Test Board and Chip Packaging

To characterize the fabricated chip, we designed a custom printed circuit board (PCB) that provides regulated power supplies, clock generation, and signal routing to the P^{2}M die. The chip is wire-bonded to a cavity package and mounted at the center of the test board. Header connectors along the board edges provide access to both analog and digital I/O signals, enabling connection to external test equipment for stimulus generation and output capture.

Fig. 6.2 shows the assembled test board with the packaged P^{2}M die. The board includes decoupling capacitors and voltage regulators to ensure clean supply delivery during measurement.

Figure 6.2: Custom test PCB with the packaged P^{2}M chip (center). The board provides regulated power, clock distribution, and signal access for both analog and digital I/O through peripheral header connectors.

6.3 Laboratory Test Setup and Status

Silicon characterization is currently underway. The test setup, shown in Fig. 6.3, interfaces the P^{2}M test board with laboratory instrumentation including a function generator for clock and control signal generation, a precision source measure unit (SMU) for current and voltage biasing, a multi-channel DC power supply, and a digital oscilloscope for capturing output waveforms. Breadboard-level signal conditioning circuits are used to adapt voltage levels

between the test equipment and the chip I/O.

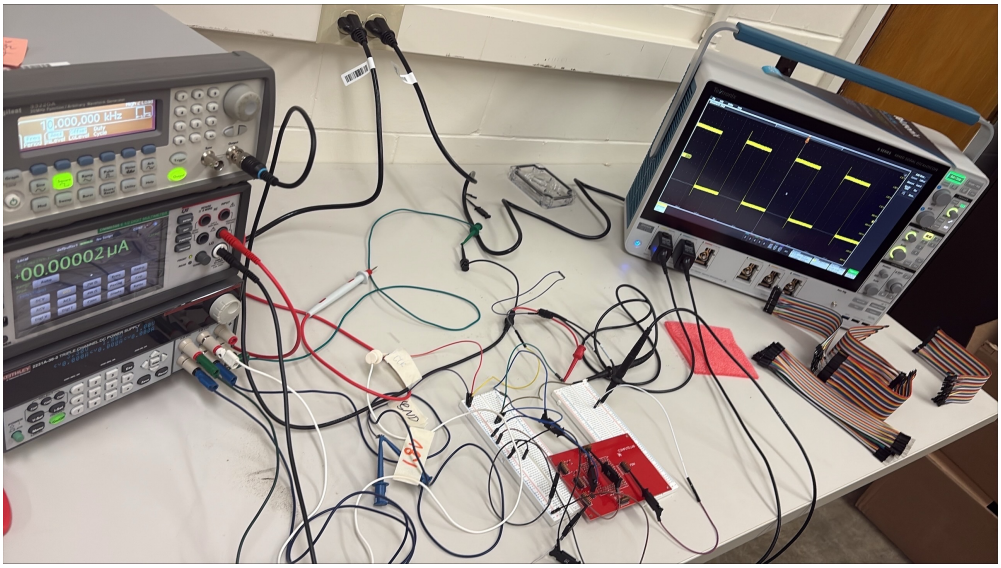


Figure 6.3: Laboratory measurement setup for P^2M chip characterization. The setup includes a function generator (top left), precision source measure unit (bottom left), DC power supply, and digital oscilloscope (right), connected to the P^2M test board (bottom right) through signal conditioning circuits.

The planned measurement campaign targets three objectives: (i) verifying the basic functionality of the weight-embedded pixel array by measuring pixel output voltage as a function of simulated photocurrent and activated weight transistor width, (ii)validating the end-to-end in-pixel convolution operation across multiple channels and comparing measured outputs against the SPICE-calibrated behavioral models used during algorithm–circuit co-design (Chapter 3). Silicon measurement results, once available, will provide the first physical validation of the P^2M paradigm and inform future iterations of the design.

Chapter 7

Conclusion and Future Work

This chapter summarizes the contributions and findings of the dissertation (§7.1), analyzes the benefits and costs across the three proposed designs (§7.2), discusses limitations and remaining challenges (§7.3), outlines future research directions (§7.4), and concludes with closing remarks (§7.5).

7.1 Conclusions

This dissertation investigated processing at the sensor front end as a system-level strategy to reduce the dominant costs of conventional sensing-to-inference pipelines: ADC conversion, data movement, and sensor-to-processor bandwidth. By mapping CNN front-end computation into the sensor, specifically into the pixel array, the sensor can output compressed feature representations rather than raw frames, enabling substantial reductions in bandwidth and improving overall energy-latency behavior.

Across three core works, the dissertation demonstrated a progression from feasibility and end-to-end benefit, to field programmability, to robustness-oriented circuit design:

- **P^{2}M (Chapter 3)** demonstrated that executing CNN front-end operations within the sensor can reduce output bandwidth by approximately 21× and improve end-to-end energy-delay product (EDP) by up to ~11× in the evaluated TinyML workload setting. P^{2}M introduced a Processing-in-Pixel-in-Memory paradigm that co-locates input activations and multi-bit weights within pixel structures, repurposes CDS and SS-ADC readout circuits for signed accumulation and batch normalization, and incorporates

an algorithm–circuit co-design methodology that models SPICE-calibrated pixel-level non-idealities during CNN training.

- **FPCA (Chapter 4)** advanced processing-in-pixel from a fixed mapped layer toward a field-programmable front-end substrate, enabling reconfiguration of key front-end parameters (kernel size, stride, and channel count) to improve deployment flexibility across workloads. FPCA introduced a compact 4T pixel architecture with hybrid CMOS–NVM weight storage and switch-matrix routing, and demonstrated the dependence of bandwidth reduction, energy, and latency on CNN hyperparameters in the TSMC 28nm HPC+ technology node.
- **CTIA-IPC (Chapter 5)** developed a robustness-oriented pathway based on CTIA pixel structures and circuit mapping innovations, demonstrating approximately 12× bandwidth reduction with only modest task-quality degradation (∼1.3–2.5% IoU on the EndoVis surgical segmentation benchmark), along with a throughput of 1.98 GOPS at an energy efficiency of 3.39 GOPS/W. The CTIA-based design leverages superior linearity and low-noise characteristics to preserve MAC fidelity under practical sensing constraints including low-light conditions.

Collectively, these results support the thesis that front-end processing in the sensor, when co-designed with readout primitives and circuit behavior, can deliver meaningful system-level savings while maintaining task performance.

7.2 Benefits and Costs Across the Three Designs

The three architectures presented in this dissertation, P^2M, FPCA, and CTIA-IPC, represent a deliberate progression in capability, each addressing limitations of its predecessor while introducing new trade-offs. Table 7.1 summarizes the key benefits and costs, and the following paragraphs discuss the design trade-offs in detail.

Design	Benefits	Costs
P^2M	First to demonstrate full CNN front-end (conv + BN + ReLU) in-pixel; ~21× bandwidth reduction; up to ~11× EDP improvement; repurposes existing CDS/SS-ADC readout	Multiple weight transistors per pixel (one per output channel) increase pixel area; weights fixed at fabrication; limited to one architecture
FPCA	Field-reconfigurable kernel size, stride, and channel count; shared weight block reduces per-pixel area; NVM enables post-fabrication weight updates	Switch-matrix routing adds control complexity; requires NVM technology (endurance, drift); 3D integration of separate weight die adds manufacturing cost
CTIA-IPC	Superior MAC linearity ($R^2 = 0.995$ vs. 0.863 for P^2M); low-noise operation enables biomedical/low-light use; digital 4-bit weights via WTC improve precision	CTIA pixel with OTA is significantly larger than 3T/4T APS; higher per-pixel power (3.26 μW vs. 0.18 μW for P^2M); digital weight storage (SRAM) requires more area per pixel

Table 7.1: Summary of benefits and costs as the in-pixel computing design advances from P^2M to FPCA to CTIA-IPC.

P^2M: area cost of in-pixel weight storage. P^2M embeds weight transistors directly within each pixel, with at least one transistor per output channel (for non-overlapping stride). For N output channels, this adds at least N transistors to each pixel (e.g., 8 transistors for 8 channels), roughly tripling the transistor count relative to a standard 3T APS pixel. While 3D heterogeneous integration places the weight transistors on a separate die beneath the photodiode array, preserving optical fill factor, the silicon area on the weight die scales linearly with the number of output channels and with the array size. Furthermore, because weight values are encoded as physical transistor widths determined at layout time, the trained model is permanently fixed after fabrication. This lack of programmability means that each new CNN configuration requires a new tapeout, limiting P^2M to applications where the target workload is known at design time.

FPCA: improved area at the cost of control complexity. FPCA addresses P^2M's programmability and area limitations by replacing per-pixel weight transistors with a shared multi-channel weight block using NVM devices, connected to the pixel array through a switch-matrix routing network. This reduces the per-pixel overhead to a single additional switch transistor (4T total per pixel), substantially lowering the area cost relative to P^2M. However, the switch-matrix routing introduces additional control complexity: the ColP (column pattern) and SM (switch matrix) control signals must be carefully orchestrated to implement the desired kernel mapping, stride, and channel configuration. The number

of control lines scales with the supported kernel and stride range, and the routing logic must be verified for each configuration. The use of NVM for weight storage also introduces technology-specific concerns, including write endurance, resistance drift, and read disturb, though these are mitigated by the low update frequency of deployed CNN weights. Overall, FPCA trades P^2M's simplicity for flexibility, and the design remains fundamentally an algorithm–hardware co-design problem: the choice of kernel size, stride, and channel count must be jointly optimized with the hardware constraints of the switch matrix and weight block.

CTIA-IPC: linearity at the cost of area and energy. CTIA-IPC improves computational fidelity by replacing the passive APS pixel with an active CTIA pixel that includes an operational transconductance amplifier (OTA), achieving dramatically better MAC linearity ($R^2 = 0.995$ versus 0.863 for APS-based P^2M). The weight-to-time converter (WTC) with 4-bit digital SRAM weights further improves precision over the analog transistor-width encoding used in P^2M. However, these improvements come at significant area and energy cost. The CTIA pixel with its OTA is substantially larger than a 3T or 4T pixel design, and the per-pixel SRAM for digital weight storage adds further area overhead ($\sim$17–18 μm^2 additional area, $< 20\%$ of the $\sim$100 μm^2 CTIA pixel footprint). The active OTA also increases per-pixel power consumption by approximately $18\times$ (3.26 μW versus 0.18 μW for P^2M), which directly impacts the total front-end energy budget. For biomedical applications with small array sizes, this power overhead is acceptable given the critical need for high-fidelity computation under low-light conditions. For large-format consumer imaging arrays, however, the energy cost of CTIA-IPC may be prohibitive, making APS-based designs (P^2M or FPCA) more appropriate despite their lower MAC precision.

Summary of the progression. The three designs illustrate a fundamental trade-off space in in-pixel computing: *area and energy cost versus computational precision and flexibility*. P^2M minimizes circuit complexity but sacrifices programmability and precision. FPCA recovers programmability with moderate control overhead. CTIA-IPC maximizes computational fidelity but at the highest area and energy cost. No single design dominates across all metrics; the appropriate choice depends on the target application's requirements for accuracy, power, programmability, and sensor format. This trade-off landscape motivates future work on hybrid architectures that selectively combine elements from each design, as well as continued

algorithm–hardware co-design to push the Pareto frontier of accuracy versus hardware cost.

7.3 Limitations and Remaining Challenges

While the dissertation establishes feasibility and value for in-pixel CNN front-end processing, several challenges remain for broader practical adoption:

1. **Variability and calibration.** Analog non-idealities such as device mismatch, temperature drift, and aging can shift transfer characteristics over time. Robust deployment in production sensors may require lightweight calibration routines, self-test modes, or training strategies that tolerate parameter drift across operating conditions.

2. **Programmability versus pixel cost.** Increasing in-pixel flexibility (supporting more kernel sizes, stride options, channel counts, and higher weight precision) competes directly with pixel area, routing complexity, and readout overhead. Future designs must carefully partition what computation belongs in-pixel, in-column, and in-periphery to balance programmability against fill factor and noise constraints.

3. **System integration.** The bandwidth reduction achieved at the sensor output must be preserved through downstream interfaces and memory hierarchies; otherwise, bottlenecks may migrate rather than disappear. Co-optimizing sensor output formats with SoC interfaces, memory controllers, and downstream inference accelerators is important for realizing end-to-end system-level benefit in product-level deployments.

7.4 Future Work

The core principle of this dissertation, compute where the signal is captured to reduce bandwidth and energy, extends naturally beyond optical image sensors. Several sensing modalities already face severe readout, bandwidth, and interconnect challenges that make them compelling targets for processing-at-the-front-end. This section discusses three future research directions: extending the approach to other sensor modalities (§7.4.1), sensor fusion as a front-end systems problem (§7.4.2), and heterogeneous integration through interposer-based architectures (§7.4.3).

7.4.1 Extending Processing-at-the-Front-End to Other Sensing Modalities

Beyond CMOS image sensors, multiple sensor types rely on dense 2D arrays where each element produces data that must be acquired and processed. In many of these modalities, the data volume from the array already exceeds what interconnects can comfortably carry, making front-end compression a system necessity rather than an optimization. The following paragraphs survey five such modalities with concrete specifications illustrating the bandwidth challenge and the opportunity for in-sensor or near-sensor front-end processing.

Ultrasound transducer arrays. Medical ultrasound probes are array sensors where each piezoelectric or capacitive micromachined ultrasonic transducer (CMUT) element generates a high-sample-rate time series that must be acquired for beamforming. Conventional 1D linear arrays typically include 128 elements (and often 192 or 256 in high-end probes), while 2D matrix arrays for volumetric imaging scale to thousands of elements [109]. At a typical sampling rate of 40 MS/s with 12-bit ADC resolution, a 128-element probe generates a peak raw data rate of approximately 61 Gbps; a 2D matrix array with over 9,000 elements (e.g., the Philips xMATRIX class) produces approximately 4.4 Tbps, far exceeding what any probe cable can carry [109]. This interconnect bottleneck has driven commercial adoption of in-probe sub-array beamforming application-specific integrated circuits (ASICs) that partially process the data at the transducer, reducing thousands of element channels to 128–256 cable lines.

Row-column addressing offers another architectural approach: by patterning orthogonal electrode strips on the top and bottom surfaces of a 2D array, $N + N = 2N$ connections replace the N^2 connections of a fully wired array [110]. For a 128×128 array, this reduces wiring from 16,384 to 256, a $64\times$ reduction. The most aggressive integration to date is the Butterfly iQ platform, which fabricates CMUT transducer membranes monolithically on a standard CMOS wafer (28 nm), placing every transducer element directly above its own analog front-end circuitry including transmit pulsing, receive amplification, ADC, and digital beamforming [111]. This monolithic CMUT-on-CMOS architecture is conceptually parallel to the in-pixel computing paradigm developed in this dissertation: computation is embedded at each sensing element to reduce data movement. Adapting the CNN front-end concept from this dissertation to ultrasound would involve implementing early operators

near the transducer array (e.g., sub-array summation, preliminary beamforming, or task-specific feature extraction) so that compressed representations, rather than raw element data, traverse the cable to the downstream processor.

Infrared focal-plane arrays. Uncooled microbolometer focal-plane arrays (FPAs) for long-wave infrared (LWIR) imaging commonly operate at formats of 320×240 to 640×512, with pixel pitches of 12–17 μm, and high-end formats reaching 1024×768 [112]. At 14-bit depth and 30–60 fps, a 640×512 FPA generates 138–275 Mbps of raw thermal data, comparable to the visible-band CIS data rates addressed in this dissertation. Cooled FPAs for tactical and scientific applications scale to 2048×2048 and beyond, producing multi-Gbps data streams [112].

A distinctive feature of microbolometer FPAs is the mandatory non-uniformity correction (NUC) step: every pixel has a unique gain and offset that drifts with temperature and must be corrected before the image is usable [113]. This per-pixel gain-and-offset correction is functionally a 1×1 convolution and is typically performed by an external digital signal processor (DSP) or field-programmable gate array (FPGA), consuming 200–500 mW. Since the in-pixel architectures developed in this dissertation (P^2M, FPCA, CTIA-IPC) already support multi-bit convolution with batch normalization (which subsumes affine per-pixel correction), NUC could be absorbed into the in-pixel compute framework as a special case. Furthermore, CTIA readout circuits are already the standard readout topology for both uncooled and cooled IR FPAs [6, 101], making the CTIA-IPC architecture directly applicable with minimal circuit adaptation. The 12–17 μm pixel pitch in IR FPAs provides substantially more silicon area per pixel than modern visible CIS (1–4 μm pitch), relaxing the transistor density constraint and making in-pixel weight storage and compute circuitry more feasible. Applications including always-on thermal surveillance, autonomous driving thermal cameras [114], and medical thermography would benefit from in-pixel feature extraction that reduces the thermal video stream by 5–20$\times$ while eliminating the external NUC processor.

Acoustic microphone arrays. Acoustic imaging systems ("acoustic cameras") use large planar arrays of microelectromechanical systems (MEMS) microphones to perform beamforming based sound source localization and mapping. Commercial products routinely deploy tens to hundreds of microphones: the Norsonic Nor848B specifies 128 MEMS mi-

crophones in a circular array, while research arrays for aeroacoustic testing exceed 1,000 elements [115]. Each MEMS microphone typically outputs a pulse-density-modulated (PDM) bitstream at 1–4 MHz clock rate. For a 128-microphone array at 48 kHz / 24-bit, the aggregate raw data rate is approximately 147 Mbps; at PDM rates (3 MHz clock), the aggregate reaches approximately 393 Mbps [115].

The front-end processing in acoustic arrays, delay-and-sum beamforming, is structurally identical to the weighted spatial summation performed by a convolution kernel: the beamformer output for a given steering direction is $y(t,\theta) = \sum_{n=0}^{N-1} w_n \, x_n(t - \tau_n(\theta))$, a weighted sum across array elements that is directly analogous to the MAC operations computed in-pixel in P^2M and FPCA [116]. After beamforming, the output (a 2D source map) is far more compact than the raw multi-channel stream, offering bandwidth reductions of $30\times$ or more, depending on the spatial resolution of the source map relative to the number of microphones. Applying the processing-at-the-front-end principle to acoustic arrays would involve performing beamforming or spatial filtering closer to (or within) the microphone array substrate, reducing the bandwidth of raw multi-channel waveforms sent to the main processor. Recent work on always-on sound event detection ASICs integrated with MEMS microphones at power levels below 15 μW [117] demonstrates the feasibility of near-microphone intelligence at extremely low power budgets.

Tactile and pressure sensor arrays. Large-area tactile sensor arrays for robotic e-skin and human-robot interaction have been demonstrated at resolutions up to 64×64 taxels (4,096 sensing sites) on flexible substrates, with taxel pitches of 1–5 mm [118, 119]. Whole-robot skin coverage distributes thousands of taxels across a robot body, with systems exceeding 4,000 sensing sites for full-body coverage of humanoid robots [120]. At scanning rates of 100 Hz to 1 kHz and 10–12 bit depth, a 64×64 array generates up to 49 Mbps of raw data; scaling to whole-body coverage at 10,000+ taxels and 1 kHz reaches 120+ Mbps.

Unlike image sensors, where the array resides on a single die, tactile skin must cover three-dimensional curved surfaces with long, flexible interconnects. Wiring complexity and connector count are often the primary scaling bottleneck rather than per-element processing cost [120]. During typical manipulation tasks, only 5–20% of the skin surface is in active contact at any moment, making dense full-frame streaming highly redundant. Event-driven readout, where each skin patch transmits data only when a tactile event (pressure change exceeding a threshold) is detected, has been shown to reduce bandwidth by up to $10\times$ in

typical scenarios [121]. Combining event-driven sensing with local spatial feature extraction (contact region detection, center-of-pressure estimation, slip features) at or near the array substrate would multiply this bandwidth savings further, following the same paradigm as the in-pixel CNN front-end developed in this dissertation.

Millimeter-wave radar arrays. Millimeter-wave (mmWave) radar is an emerging sensing modality for applications including autonomous driving, gesture recognition, and human pose estimation. mmWave radar arrays typically operate at 60–77 GHz with bandwidths of 1–4 GHz, producing high-dimensional range-Doppler-angle data cubes that require substantial processing. A radar array with 12 receive antennas at 10 Msps and 16-bit ADC resolution generates over 1.9 Gbps of raw intermediate-frequency (IF) data per frame, and multi-input multi-output (MIMO) configurations with virtual arrays exceeding 192 elements scale proportionally. Recent work has demonstrated multi-modal 3D human pose estimation by fusing mmWave radar with RGB-D and inertial sensors [122], highlighting the rich spatial information available from mmWave arrays and the potential for front-end feature extraction to reduce the data volume before fusion. The signal processing pipeline for mmWave radar (range FFT, Doppler FFT, angle estimation) involves weighted spatial summation operations structurally similar to those targeted by in-pixel computing, suggesting that processing-at-the-front-end could be adapted to radar arrays to compress range-Doppler-angle representations near the antenna elements.

Summary of the cross-modality direction. Across all five modalities, the generalized thesis principle is: replace dense raw-array streaming with sensor-side computed, compressed representations (features, events, or regions of interest), while accounting for the modality-specific analog front-end physics and noise/linearity requirements. Table 7.2 summarizes the key characteristics and bandwidth challenges of each modality.

Modality	Typical Array	Raw Bandwidth	Key Bottleneck	Front-End Operation
Visible CIS (this work)	560×560 RGB	~395 Mbps	ADC + I/O energy	CNN convolution
Ultrasound	128–9,000+ el.	61 Gbps – 4.4 Tbps	Probe cable	Beamforming
IR FPA	640×512	138–275 Mbps	NUC power + I/O	NUC + feature extraction
Acoustic array	128–1,024 mics	147 Mbps – 3+ Gbps	Multi-ch. streaming	Beamforming
Tactile array	64×64+	49–120+ Mbps	Wiring / connectors	Event + spatial filtering
mmWave radar	12–192+ el.	1.9+ Gbps	IF data volume	Range-Doppler-angle compression

Table 7.2: Summary of array sensor modalities where processing-at-the-front-end can reduce data movement. Raw bandwidth is computed at representative operating points (see text for assumptions).

7.4.2 Sensor Fusion as a Front-End Systems Problem

Sensor fusion combines data from multiple modalities to improve robustness, reduce ambiguity, or enable capabilities not achievable with a single sensor [114]. The practical challenge is that naive multi-sensor fusion often multiplies system bandwidth and compute requirements by streaming multiple dense data sources to a central processor, a problem that can be termed "multi-sensor bandwidth explosion." Processing-at-the-front-end offers a path to address this: rather than streaming raw data from every modality, each sensor performs modality-appropriate front-end compression and the compressed outputs are designed to be efficiently combined downstream.

A key insight is that not all sensors in a fused system need to operate at full bandwidth simultaneously. In many fusion scenarios, one low-power modality can gate or condition the operation of a higher-power modality, so that the expensive sensor activates only when a relevant event is detected. This *event-gated fusion* pattern reduces the average system power and bandwidth without sacrificing responsiveness, and can be implemented at the sensor front end or at a shared interposer layer (Section 7.4.3).

Table 7.3 illustrates four concrete fusion scenarios that map naturally onto the processing-at-the-front-end paradigm developed in this dissertation.

Fusion Type	Application	Fusion Mechanism	Hardware Benefit
Acoustic + IR	Gesture/voice control	IR array detects human presence; triggers wake-up of acoustic processing chain	Energy gating: high-power audio DSP activates only on presence detection, reducing average system power
Pressure + Ultrasound	Medical imaging	Pressure sensor array confirms correct probe contact force before activating ultrasound transmit pulses	Conditional activation: ultrasound front-end (beamforming, ADC) powers on only when contact quality is sufficient, avoiding wasted energy on invalid acquisitions
Optical + Acoustic	Autonomous navigation	Camera detects vehicle; microphone array detects siren; interposer-level logic correlates both to trigger immediate response	Early fusion at interposer: lightweight correlation logic between compressed sensor outputs enables low-latency decision (e.g., emergency stop) without full-frame transfer to main SoC
mmWave + Camera	3D pose estimation	mmWave radar provides depth and velocity; camera provides appearance and fine-grained spatial features [122]	Complementary fusion: radar operates through occlusion/low-light where camera fails; front-end compression on both modalities reduces aggregate bandwidth

Table 7.3: Representative multi-modal sensor fusion scenarios amenable to front-end processing. In each case, one modality conditions or gates the other, enabling hardware-level energy and bandwidth savings through event-driven or conditional activation.

Acoustic + IR for gesture and voice control. In smart-home or wearable devices, an infrared thermal sensor array (e.g., a low-resolution microbolometer FPA as discussed in Section 7.4.1) can serve as an always-on presence detector at microwatt-level power. When the IR front end detects a thermal signature consistent with a human user, it issues a wake signal to the acoustic processing chain, which then activates higher-power MEMS microphone beamforming and keyword spotting. Without this gating, the acoustic subsystem would run continuously, consuming tens of milliwatts for always-on listening. The front-

end processing principle applies directly: the IR sensor outputs a binary or low-bit event ("presence detected") rather than a full thermal frame, and the acoustic array activates its beamforming front end only on demand.

Pressure + ultrasound for medical imaging. In handheld ultrasound probes, image quality depends critically on maintaining proper acoustic coupling between the transducer and the patient's skin, which requires adequate and uniform contact pressure. A pressure sensor array integrated into the probe face can monitor contact force distribution in real time. Rather than running the full ultrasound transmit-receive-beamform pipeline continuously, the system activates ultrasound acquisition only when the pressure front end confirms that contact quality meets a threshold. This conditional activation avoids wasting energy on acquisitions that would yield unusable images due to air gaps or poor coupling, and reduces average power in battery-operated point-of-care devices. The pressure sensor itself requires minimal bandwidth (a few hundred bytes per frame), making the gating overhead negligible.

Optical + acoustic for autonomous navigation. In autonomous driving, fusing visual and acoustic information can enable rapid detection of emergency vehicles. A camera identifies a vehicle in the visual field using the in-pixel CNN front end, while a microphone array performs beamforming-based sound source localization to detect a siren. When both modalities independently detect correlated evidence (vehicle + siren from the same direction), a lightweight correlation primitive at the interposer or package level can trigger an immediate response signal without waiting for the full data from either sensor to reach the central SoC. This "early fusion at the edge" pattern exploits the compressed feature outputs from both sensor front ends (visual feature maps and acoustic source-direction estimates) to achieve low-latency decision-making with minimal data transfer.

7.4.3 Interposer and Heterogeneous Integration for Pre-Processing and Early Fusion

A practical hardware path for multi-sensor systems is heterogeneous integration, including 2.5D interposers, where multiple dies (sensors, memory, compute) are co-packaged with high-bandwidth, low-energy interconnects [123]. Recent work on chiplet-based architectures for deep learning has demonstrated co-simulation frameworks for multi-chiplet

systems [124] and energy-efficient heterogeneous multi-chiplet DNN inference [125], while thermally-aware scheduling of AI workloads on heterogeneous processing-in-memory chiplet architectures has been explored to manage the thermal challenges of tightly integrated compute [126]. These advances in chiplet-based design methodology directly support the multi-sensor heterogeneous integration vision proposed here. Silicon interposers with through-silicon vias (TSVs) at 5–10 μm pitch are already in production for high-bandwidth memory stacking [127], and Cu-Cu hybrid bonding at sub-5 μm pitch has been demonstrated for stacked CMOS image sensors [84, 128]. Die-to-die links through a silicon interposer consume approximately 0.3–1.0 pJ/bit, compared to 5–20 pJ/bit for conventional off-chip I/O [123, 129], representing a 10–50$\times$ reduction in per-bit communication energy.

This dissertation already leverages 3D integration for in-pixel weight storage: both FPCA and CTIA-IPC use vertically stacked SRAM or non-volatile memory (NVM) dies bonded to the pixel array via TSV or Cu-Cu bonding. The natural extension is to move from single-sensor 3D stacking to multi-sensor 2.5D or 3D heterogeneous integration. An interposer or package substrate layer could host: (i) shared weight storage close to multiple sensors, (ii) lightweight front-end compute blocks for early convolution, beamforming, or filtering, and (iii) early fusion primitives such as temporal alignment, gating logic, and low-cost feature fusion. With this architecture, only compressed and fused outputs would be sent to the main SoC, potentially compounding the bandwidth and energy savings demonstrated in this dissertation for single sensors.

This direction is especially compelling for modalities with severe wiring or bandwidth constraints. In ultrasound probes, for example, integrated electronics already reduce cabling from thousands of element connections to 128–256 system channels [109]; an interposer hosting beamforming compute alongside the transducer array could push this compression further. Similarly, neural recording arrays such as Neuropixels [130] face stringent power and data budgets for implantable operation, where heterogeneous co-packaging of recording, processing, and communication dies through short-reach interposer links could enable compressed sensing or early feature extraction at the probe, analogous to the in-pixel CNN front-end developed in this work.

This direction can be made quantitative by extending the bandwidth reduction (BR) and energy-delay product (EDP) methodology developed in Chapters 3–5 to multi-sensor stacks: measuring (i) raw aggregate bandwidth versus compressed/fused bandwidth, (ii) interposer compute energy versus saved I/O and SoC energy, and (iii) task-level robustness gains from

multi-modal fusion. Such a framework would provide a principled basis for evaluating the system-level benefits of heterogeneous front-end integration across sensor modalities.

7.5 Closing Remarks

Processing-at-the-front-end is a broad architectural principle: when computation is moved to the point of signal acquisition, systems can reduce data movement, improve latency, and enable always-on intelligence under tight energy budgets. The contributions of this dissertation establish feasibility and offer practical design pathways for CMOS image sensors, from end-to-end system benefit (P^2M) through field-programmable deployment (FPCA) to robustness-oriented circuit design (CTIA-IPC). The future directions discussed above extend the same principles to broader sensing modalities, fusion-centric multi-sensor systems, and packaging-enabled architectures that can amplify the system-level benefits beyond standalone image sensors.

Bibliography

[1] Aaron Stillmaker and Bevan Baas. Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm. *Integration*, 58:74–81, 2017.

[2] Mustafa Ali, Akhilesh Jaiswal, Sangamesh Kodge, Amogh Agrawal, Indranil Chakraborty, and Kaushik Roy. IMAC: In-memory multi-bit multiplication and accumulation in 6T sram array. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(8):2521–2531, 2020.

[3] E.R. Fossum. Cmos image sensors: electronic camera-on-a-chip. *IEEE Transactions on Electron Devices*, 44(10):1689–1698, 1997.

[4] Takayuki Toyama, Koji Mishina, Hiroyuki Tsuchiya, Tatsuya Ichikawa, Hiroyuki Iwaki, Yuji Gendai, Hirotaka Murakami, Kenichi Takamiya, Hiroshi Shiroshita, Yoshinori Muramatsu, et al. A 17.7 mpixel 120fps cmos image sensor with 34.8 gb/s readout. In *2011 IEEE International Solid-State Circuits Conference*, pages 420–422. IEEE, 2011.

[5] Mei Zou, Nan Chen, Yue-sheng Pu, and Li-bin Yao. A low-light-level cmos image sensor with a novel correlated double sampling based on ctia. *Microelectronics Journal*, 150:106262, 2024.

[6] Yi Zhuo, Wengao Lu, Shanzhe Yu, Ye Zhou, Jiaqi Kong, Yacong Zhang, and Zhongjian Chen. A low-noise ctia-based pixel with cds for swir focal plane arrays. In *2021 6th International Conference on Integrated Circuits and Microsystems (ICICM)*, pages 258–262, 2021.

[7] Kyuik Cho, Daeyun Kim, and Minkyu Song. A low power dual cds for a column-parallel cmos image sensor. *JSTS:Journal of Semiconductor Technology and Science*, 12, 12 2012.

[8] Jiaju Ma, Saleh Masoodian, Dakota A. Starkey, and Eric R. Fossum. Photon-number-resolving megapixel image sensor at room temperature without avalanche gain. *Optica*, 4(12):1474–1481, Dec 2017.

[9] Reid Pinkham, Andrew Berkovich, and Zhengya Zhang. Near-sensor distributed dnn processing for augmented and virtual reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):663–676, 2021.

[10] Sony to Release World's First Intelligent Vision Sensors with AI Processing Functionality. `https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/`, 2020. Accessed: 12-01-2022.

[11] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. Reconfiguring the imaging pipeline for computer vision. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 975–984, 2017.

[12] Zhe Chen et al. Processing near sensor architecture in mixed-signal domain with cmos image sensor of convolutional-kernel-readout method. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(2):389–400, 2020.

[13] Lukas Mennel et al. Ultrafast machine vision with 2D material neural network image sensors. *Nature*, 579:62–66, 2020.

[14] Laurie Bose et al. Fully embedding fast convolutional networks on pixel processor arrays. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374, pages 488–503. Springer, 2020.

[15] Ruibing Song, Kejie Huang, Zongsheng Wang, and Haibin Shen. A reconfigurable convolution-in-pixel cmos image sensor architecture. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10):7212–7225, 2022.

[16] Venkatesh Kodukula et al. Dynamic temperature management of near-sensor processing for energy-efficient high-fidelity imaging. *Sensors*, 1(3), 2021.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] Manjunath Jogin, Mohana, M S Madhulika, G D Divya, R K Meghana, and S Apoorva. Feature extraction using convolution neural networks (CNN) and deep learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, volume 1, pages 2319–2323, 2018.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[25] ON Semiconductor. *CMOS Image Sensor, 1.2 MP, Global Shutter*, 3 220. Rev. 10.

[26] Shubham Jain, Abhronil Sengupta, Kaushik Roy, and Anand Raghunathan. RxNN: A framework for evaluating deep neural networks on resistive crossbars. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 40(2):326–338, feb 2021.

[27] Corey Lammie and Mostafa Rahimi Azghadi. Memtorch: A simulation framework for deep memristive cross-bar architectures. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 1, pages 1–5, 2020.

[28] M Lefebvre, L Moreau, R Dekimpe, and D Bol. 7.7 a 0.2-to-3.6 tops/w programmable convolutional imager soc with in-sensor current-domain ternary-weighted mac operations for feature extraction and region-of-interest detection. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 118–120. IEEE, 2021.

[29] T Hsu, Y Chen, R Liu, C Lo, K Tang, M Chang, and C Hsieh. A 0.5-v real-time computational cmos image sensor with programmable kernel for feature extraction. *IEEE Journal of Solid-State Circuits*, 56(5):1588–1596, 2020.

[30] R Song, K Huang, Z Wang, and H Shen. A reconfigurable convolution-in-pixel cmos image sensor architecture. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10):7212–7225, 2022.

[31] Arman Roohi, Sepehr Tabrizchi, Mehrdad Morsali, David Z Pan, and Shaahin Angizi. Pipsim: A behavior-level modeling tool for cnn processing-in-pixel accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(1):141–150, 2023.

[32] Jiyang Xie, Yixiao Zheng, Ruoyi Du, Weiyu Xiong, Yufei Cao, Zhanyu Ma, Dongpu Cao, and Jun Guo. Deep learning-based computer vision for surveillance in its: Evaluation of state-of-the-art methods. *IEEE Transactions on Vehicular Technology*, 70(4):3027–3042, 2021.

[33] Umair Iqbal, Pascal Perez, Wanqing Li, and Johan Barthelemy. How computer vision can facilitate flood management: A systematic review. *International Journal of Disaster Risk Reduction*, 53:102030, 2021.

[34] Alexander Gomez, Augusto Salazar, and Francisco Vargas. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169*, 2016.

[35] Scaling CMOS Image Sensors. `https://semiengineering.com/scaling-cmos-image-sensors/`, 2020. Accessed: 04-20-2020.

[36] Terrence J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

[37] Akhilesh Jaiswal and Ajey Jacob. Integrated pixel and two-terminal non-volatile memory cell and an array of cells for deep in-sensor, in-memory computing, December 7 2021. US Patent 11,195,580.

[38] Akhilesh Jaiswal and Ajey Jacob. Integrated pixel and three-terminal non-volatile memory cell and an array of cells for deep in-sensor, in-memory computing, July 20 2021. US Patent 11,069,402.

[39] Shaahin Angizi et al. PISA: A binary-weight processing-in-sensor accelerator for edge image processing. *arXiv preprint arXiv:2202.09035*, 2022.

[40] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[41] Guilian Gao, Thomas Workman, Laura Mirkarimi, Gill Fountain, Jeremy Theil, Gabe Guevara, Cyprian Uzoh, Bongsub Lee, Ping Liu, and Pawel Mrozek. Chip to wafer hybrid bonding with Cu interconnect: High volume manufacturing process compatibility study. In *2019 International Wafer Level Packaging Conference (IWLPC)*, volume 1, pages 1–9, 2019.

[42] Vincent C. Venezia, Alan Chih-Wei Hsiung, Kelvin Ai, Xiang Zhao, Zhiqiang Lin, Duli Mao, Armin Yazdani, Eric A. G. Webster, and Lindsay A. Grant. 1.5µm dual conversion gain, backside illuminated image sensor using stacked pixel level connections with 13ke-full-well capacitance and 0.8e-noise. In *2018 IEEE International Electron Devices Meeting (IEDM)*, volume 1, pages 10.1.1–10.1.4, 2018.

[43] Shunichi Sukegawa, Taku Umebayashi, Tsutomu Nakajima, Hiroshi Kawanobe, Ken Koseki, Isao Hirota, Tsutomu Haruta, Masanori Kasai, Koji Fukumoto, Toshifumi Wakano, Keishi Inoue, Hiroshi Takahashi, Takashi Nagano, Yoshikazu Nitta, Teruo Hirayama, and Noriyuki Fukushima. A 1/4-inch 8Mpixel back-illuminated stacked CMOS image sensor. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, volume 1, pages 484–485, 2013.

[44] Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger. Phase-change technology and the future of main memory. *IEEE Micro*, 30(1):143–143, 2010.

[45] Kaiyuan Guo, Jincheng Yu, Xuefei Ning, Yiming Hu, Yu Wang, and Huazhong Yang. RRAM based buffer design for energy efficient cnn accelerator. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, volume 1, pages 435–440, 2018.

[46] Yu-Der Chih, Yi-Chun Shih, Chia-Fu Lee, Yen-An Chang, Po-Hao Lee, Hon-Jarn Lin, Yu-Lin Chen, Chieh-Pu Lo, Meng-Chun Shih, Kuei-Hung Shen, Harry Chuang, and Tsung-Yung Jonathan Chang. 13.3 a 22nm 32Mb embedded STT-MRAM with 10ns read speed, 1M cycle write endurance, 10 years retention at 150°c and high immunity to magnetic field interference. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, volume 1, pages 222–224, 2020.

[47] Asif Khan, Ali Keshavarzi, and S. Datta. The future of ferroelectric field-effect transistor technology. *Nature Electronics*, 3:588–597, 10 2020.

[48] M Gupta et al. High-density SOT-MRAM technology and design specifications for the embedded domain at 5nm node. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 24–5, 2020.

[49] Oindrila Saha et al. RNNPool: Efficient non-linear pooling for RAM constrained inference. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20473–20484, 2020.

[50] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[51] Partha Pratim Ray. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 2021.

[52] Bharath Sudharsan, Simone Salerno, Duc-Duy Nguyen, Muhammad Yahya, Abdul Wahid, Piyush Yadav, John G. Breslin, and Muhammad Intizar Ali. TinyML benchmark: Executing fully connected neural networks on commodity microcontrollers. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, volume 1, pages 883–884, 2021.

[53] Colby Banbury, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. Micronets: Neural network architectures for deploying TinyML applications on commodity microcontrollers. In A. Smola, A. Dimakis, and I. Stoica, editors, *Proceedings of Machine Learning and Systems*, volume 3, pages 517–532, 2021.

[54] Aakanksha Chowdhery, Pete Warden, Jonathon Shlens, Andrew Howard, and Rocky Rhodes. Visual wake words dataset. *arXiv preprint arXiv:1906.05721*, 2019.

[55] Meet Astro, a home robot unlike any other. `https://www.aboutamazon.com/news/devices/meet-astro-a-home-robot-unlike-any-other`, 2021. Accessed: 09-28-2021.

[56] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[57] Colby R. Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, David Patterson, Danilo Pau, Jae-sun Seo, Jeff Sieracki, Urmish Thakker, Marian Verhelst, and Poonam Yadav. Benchmarking tinyml systems: Challenges and direction. *arXiv preprint arXiv:2003.04821*, 2020.

[58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2015.

[59] Song Han, Ji Lin, Kuan Wang, Tianze Wang, and Zhanghao Wu. Solution to visual wakeup words challenge'19 (first place). `https://github.com/mit-han-lab/VWW`, 2019.

[60] Chuteng Zhou, Fernando Garcia Redondo, Julian Büchel, Irem Boybat, Xavier Timoneda Comas, S. R. Nandakumar, Shidhartha Das, Abu Sebastian, Manuel Le Gallo, and Paul N. Whatmough. Analognets: ML-HW co-design of noise-robust TinyML models and always-on analog compute-in-memory accelerator. *arXiv preprint arXiv:2111.06503*, 2021.

[61] Souvik Kundu et al. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*, 69(7):1045–1058, 2020.

[62] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A. Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3953–3962, January 2021.

[63] Gourav Datta, Souvik Kundu, and Peter A. Beerel. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding. In *2021 International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 1–8, 2021.

[64] Gourav Datta and Peter A. Beerel. Can deep neural networks be converted to ultra low-latency spiking neural networks? *arXiv preprint arXiv:2112.12133*, 2021.

[65] Souvik Kundu et al. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021.

[66] Mingu Kang, Sungmin Lim, Sujan Gonugondla, and Naresh R Shanbhag. An in-memory vlsi architecture for convolutional neural networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(3):494–505, 2018.

[67] Gourav Datta, Souvik Kundu, Akhilesh Jaiswal, and Peter A. Beerel. HYPER-SNN: Towards Energy-efficient Quantized Deep Spiking Neural Networks for Hyperspectral Image Classification. *arXiv preprint arXiv:2107.11979*, July 2021.

[68] Mohammad Faisal Amir and S. Mukhopadhyay. 3D stacked high throughput pixel parallel image sensor with integrated ReRAM based neural accelerator. *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pages 1–3, 2018.

[69] Y Chen, H Dai, and Y Ding. Pseudo-stereo for monocular 3d object detection in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 887–897, 2022.

[70] L Jiao, R Zhang, F Liu, S Yang, B Hou, L Li, and X Tang. New generation deep learning for video object detection: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[71] R Eki, S Yamada, H Ozawa, H Kai, K Okuike, H Gowtham, H Nakanishi, E Almog, Y Livne, G Yuval, et al. 9.6 a 1/2.3 inch 12.3 mpixel with on-chip 4.97 tops/w cnn processor back-illuminated stacked cmos image sensor. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 154–156. IEEE, 2021.

[72] F Zhou and Y Chai. Near-sensor and in-sensor computing. *Nature Electronics*, 3(11):664–671, 2020.

[73] H Xu, N Lin, L Luo, Q Wei, R Wang, C Zhuo, X Yin, F Qiao, and H Yang. Senputing: An ultra-low-power always-on vision perception chip featuring the deep fusion of sensing and computing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(1):232–243, 2021.

[74] L Bose, P Dudek, Stephen J Chen, Jand C, and Walterio W Mayol-Cuevas. Fully embedding fast convolutional networks on pixel processor arrays. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 488–503. Springer, 2020.

[75] G Datta, S Kundu, Z Yin, J Lakkireddy, R Mathai, A Jacob, P Beerel, and A Jaiswal. A processing-in-pixel-in-memory paradigm for resource-constrained tinyml applications. *Scientific Reports*, 12(1):14396, 2022.

[76] G Datta, S Kundu, Z Yin, J Mathai, Z Liu, Z Wang, M Tian, S Lu, R Lakkireddy, et al. P 2 m-detrack: Processing-in-pixel-in-memory for energy-efficient and real-time multi-object detection and tracking. In *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2022.

[77] Zihan Yin, Gourav Datta, Md Abdullah-Al Kaiser, Peter Beerel, Ajey Jacob, and Akhilesh Jaiswal. Design considerations for 3d heterogeneous integration driven analog processing-in-pixel for extreme-edge intelligence. In *2023 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–5. IEEE, 2023.

[78] Md Abdullah-Al Kaiser, Gourav Datta, Sreetama Sarkar, Souvik Kundu, Zihan Yin, Manas Garg, Ajey P Jacob, Peter A Beerel, and Akhilesh R Jaiswal. Technology-circuit-algorithm tri-design for processing-in-pixel-in-memory (p2m). In *Proceedings of the Great Lakes Symposium on VLSI 2023*, pages 613–618, 2023.

[79] F Yu, H Chen, X Wang, W Xian, F Chen, Yand Liu, V Madhavan, and T Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[80] Akhilesh Jaiswal and Ajey Poovannummoottil Jacob. Integrated pixel and three-terminal non-volatile memory cell and an array of cells for deep in-sensor, in-memory computing, July 20 2021. US Patent 11,069,402.

[81] H Nam and B Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.

[82] Gourav Datta et al. ACE-SNN: Algorithm-hardware co-design of energy-efficient & low-latency deep spiking neural networks for 3d image recognition. *Frontiers in neuroscience*, page 400, 2022.

[83] M Seo, M Chu, H Jung, S Kim, J Song, J Lee, S Kim, J Lee, S Byun, D Bae, et al. A 2.6 e-rms low-random-noise, 116.2 mw low-power 2-mp global shutter cmos image sensor with pixel-level adc and in-pixel memory. In *2021 Symposium on VLSI Technology*, pages 1–2. IEEE, 2021.

[84] Y Kagawa, N Fujii, K Aoyagi, Y Kobayashi, S Nishi, N Todaka, S Takeshita, J Taura, H Takahashi, Y Nishimura, et al. Novel stacked cmos image sensor with advanced cu2cu hybrid bonding. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 8–4. IEEE, 2016.

[85] S Tabrizchi, A Nezhadi, S Angizi, and A Roohi. Appcip: Energy-efficient approximate convolution-in-pixel scheme for neural network acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[86] Gokul Krishnan, Jingbo Sun, Jubin Hazra, Xiaocong Du, Maximilian Liehr, Zheng Li, Karsten Beckmann, Rajiv V. Joshi, Nathaniel C. Cady, and Yu Cao. Robust rram-based in-memory computing in light of model stability. In *2021 IEEE International Reliability Physics Symposium (IRPS)*, pages 1–5, 2021.

[87] Sumit Diware, Abhairaj Singh, Anteneh Gebregiorgis, Rajiv V. Joshi, Said Hamdioui, and Rajendra Bishnoi. Accurate and energy-efficient bit-slicing for rram-based neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(1):164–177, 2023.

[88] Keming Fan, Ashkan Moradifirouzabadi, Xiangjin Wu, Zheyu Li, Flavio Ponzina, Anton Persson, Eric Pop, Tajana Rosing, and Mingu Kang. Specpcm: A low-power pcm-based in-memory computing accelerator for full-stack mass spectrometry analysis. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 10:161–169, 2024.

[89] MRH Mohd Adnan, Azlan Mohd Zain, Habibollah Haron, Mohd Zulfaezal Che Azemin, and Mahadi Bahari. Consideration of canny edge detection for eye redness image processing: A review. In *IOP Conference Series: Materials Science and Engineering*, volume 551, page 012045. IOP Publishing, 2019.

[90] S Deepa and V Subbiah Bharathi. Efficient roi segmentation of digital mammogram images using otsu's n thresholding method. *International Journal of Engineering Research & Technology*, 2(1):1–6, 2013.

[91] P Shanmuga Sundaram and N Santhiyakumari. An enhancement of computer aided approach for colon cancer detection in wce images using roi based color histogram and svm2. *Journal of medical systems*, 43(2):29, 2019.

[92] Yingcong Shi, Jiaxin Qiao, Jian Song, Wei Huang, Kai Sun, Mingxin Zhao, and Junchao Ma. Roi detection of hand bone based on yolo v3. In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 234–238. IEEE, 2021.

[93] M Suchetha, N Sai Ganesh, Rajiv Raman, and D Edwin Dhas. Region of interest-based predictive algorithm for subretinal hemorrhage detection using faster r-cnn. *Soft Computing*, 25(24):15255–15268, 2021.

[94] Gokul Krishnan, Zhenyu Wang, Injune Yeo, Li Yang, Jian Meng, Maximilian Liehr, Rajiv V. Joshi, Nathaniel C. Cady, Deliang Fan, Jae-Sun Seo, and Yu Cao. Hybrid rram/sram in-memory computing for robust dnn acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):4241–4252, 2022.

[95] Zhuoran Song, Yanan Sun, Lerong Chen, Tianjian Li, Naifeng Jing, Xiaoyao Liang, and Li Jiang. Itt-rna: Imperfection tolerable training for rram-crossbar-based deep neural-network accelerator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(1):129–142, 2021.

[96] Han Xu, Ningchao Lin, Li Luo, Qi Wei, Runsheng Wang, Cheng Zhuo, Xunzhao Yin, Fei Qiao, and Huazhong Yang. Senputing: An ultra-low-power always-on vision perception chip featuring the deep fusion of sensing and computing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(1):232–243, 2021.

[97] Muhannad S. Bakir, Madison Manley, Ashita Victor, Zhonghao Zhang, Euichul Chung, Shane Oh, and Geyu Yan. The system revolution enabled by 2.5d and 3d ic technologies. In *IEDM 2024 Short Course: Technology Innovations Shaping the Roadmap in the Era of AI (SC1.4)*, San Francisco, CA, 2024. Short Course Presentation.

[98] K Teja et al. Design of 1.8 v lvds transmitter in gf 22nm for associative memory. In *2021 International Semiconductor Conference (CAS)*, pages 201–204. IEEE, 2021.

[99] Min-Woong Seo, Myunglae Chu, Hyun-Yong Jung, Suksan Kim, Jiyoun Song, Junan Lee, Sung-Yong Kim, Jongyeon Lee, Sung-Jae Byun, Daehee Bae, Minkyung Kim, Gwi-Deok Lee, Heesung Shim, Changyong Um, Changhwa Kim, In-Gyu Baek, Doowon Kwon, Hongki Kim, Hyuksoon Choi, Jonghyun Go, JungChak Ahn, Jaekyu Lee, Changrok Moon, Kyupil Lee, and Hyoung-Sub Kim. A 2.6 e-rms low-random-noise, 116.2 mw low-power 2-mp global shutter cmos image sensor with pixel-level adc and in-pixel memory. In *2021 Symposium on VLSI Circuits*, pages 1–2, 2021.

[100] Y. Kagawa, S. Hida, Y. Kobayashi, K. Takahashi, S. Miyanomae, M. Kawamura, H. Kawashima, H. Yamagishi, T. Hirano, K. Tatani, H. Nakayama, K. Ohno, H. Iwamoto, and S. Kadomura. The scaling of cu-cu hybrid bonding for high density 3d chip stacking. In *2019 Electron Devices Technology and Manufacturing Conference (EDTM)*, pages 297–299, 2019.

[101] Qifa Zhang, Huanhui Zhang, Zhe Wang, Runjiang Dou, Liyuan Liu, Zhao Zhang, Jian Liu, Nanjian Wu, and Shushen Li. A 20μm pixel-pitch, 60me-droic based on ctia for 640 x 512 infrared focal plane array. In *2023 3rd International Conference on Electronic Information Engineering and Computer Communication (EIECC)*, pages 209–213. IEEE, 2023.

[102] Kartikeya Murari, Ralph Etienne-Cummings, Nitish V. Thakor, and Gert Cauwenberghs. A cmos in-pixel ctia high-sensitivity fluorescence imager. *IEEE Transactions on Biomedical Circuits and Systems*, 5(5):449–458, 2011.

[103] Max Allan, Alexey Shvets, Thomas Kurmann, Zichen Zhang, Rahul Duggal, Yun-Hsuan Su, Nicola Rieke, Iro Laina, Niveditha Kalavakonda, Sebastian Bodenstedt, Luis Herrera, Wenqi Li, Vladimir Iglovikov, Huoling Luo, Jian Yang, Danail Stoyanov, Lena Maier-Hein, Stefanie Speidel, and Mahdi Azizian. 2017 robotic instrument segmentation challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 135–143, 2017.

[104] Kanglin Xiao, Xin Qiao, Xiaoxin Cui, Jiahao Song, Haoyang Luo, Xin'an Wang, and Yuan Wang. A 28nm 8kb reconfigurable sram computing-in-memory macro with input-sparsity optimized dtc for multi-mode mac operations. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 71(7):3263–3267, 2024.

[105] BS Haran, A Kumar, L Adam, J Chang, V Basker, S Kanakasabapathy, D Horak, S Fan, J Chen, J Faltermeier, et al. 22 nm technology compatible fully functional 0.1 μm 2 6t-sram cell. In *2008 IEEE International Electron Devices Meeting*, pages 1–4. IEEE, 2008.

[106] Cleveland Clinic. Minimally invasive surgery: What it is, types, benefits & risks. *Cleveland Clinic*, 2023.

[107] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation. *CoRR*, abs/1801.05746, 2018.

[108] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[109] Bernard Savord and Rod Solomon. Fully sampled matrix transducer for real time 3D ultrasonic imaging. In *Proc. IEEE Ultrasonics Symposium*, volume 1, pages 945–953. IEEE, 2003.

[110] Morten Fischer Rasmussen, Thomas Lehrmann Christiansen, Martin Christian Hemmsen, and Jørgen Arendt Jensen. 3-D imaging using row–column-addressed arrays with integrated apodization—Part I: Apodization design and line element beamforming. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 62(5):947–958, 2015.

[111] Jonathan M Rothberg, Tyler S Ralston, Andrew G Rothberg, Jeffry Martin, Jaime S Zahorian, Susan A Alie, Thomas Altmann, Robert Broze, Kevin Havens, Amin Kapur, et al. Ultrasound-on-chip platform for medical imaging, analysis, and collective intelligence. *Proceedings of the National Academy of Sciences*, 118(27):e2019339118, 2021.

[112] Antoni Rogalski, Piotr Martyniuk, and Małgorzata Kopytko. Challenges of small-pixel infrared detectors: a review. *Reports on Progress in Physics*, 80(4):046501, 2017.

[113] Sergio N Torres, Jorge E Pezoa, and Majeed M Hayat. Scene-based nonuniformity correction for focal plane arrays by the method of the inverse covariance form. *Applied Optics*, 42(29):5872–5881, 2003.

[114] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1037–1045, 2015.

[115] Paolo Chiariotti, Milena Martarelli, and Paolo Castellini. Acoustic beamforming for noise source localization – reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 120:422–448, 2019.

[116] Harry L Van Trees. *Optimum Array Processing (Detection, Estimation, and Modulation Theory, Part IV)*. Wiley, 2002.

[117] S Liu et al. An always-on sound event detection MEMS microphone ASIC with 14 μW keyword spotting. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 352–354, 2023.

[118] Ravinder S Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile sensing – from humans to humanoids. *IEEE Transactions on Robotics*, 26(1):1–20, 2010.

[119] Subramanian Sundaram, Petr Kellnhofer, Yunzhu Li, Jun-Yan Zhu, Antonio Torralba, and Wojciech Matusik. Learning the signatures of the human grasp using a scalable tactile glove. *Nature*, 569:698–702, 2019.

[120] Gordon Cheng, Emmanuel Dean-Leon, Florian Bergner, Julio Rogelio Olvera, Quentin Leboutet, and Philipp Mittendorfer. A comprehensive realization of robot skin: Sensors, sensing, control, and applications. *Proceedings of the IEEE*, 107(10):2034–2051, 2019.

[121] Chiara Bartolozzi, Lorenzo Natale, Francesco Nori, and Giorgio Metta. Robots with a sense of touch. *Nature Materials*, 15:921–925, 2016.

[122] Sizhe An, Yin Li, and Umit Ogras. MRI: Multi-modal 3D human pose estimation dataset using mmwave, RGB-D, and inertial sensors. *Advances in Neural Information Processing Systems*, 35:27414–27426, 2022.

[123] Y Zhang, X Zhang, and M S Bakir. Benchmarking digital die-to-die channels in 2.5-d and 3-d heterogeneous integration platforms. *IEEE Transactions on Electron Devices*, 65(12):5460–5467, 2018.

[124] Lukas Pfromm et al. CHIPSIM: A co-simulation framework for deep learning on chiplet-based systems. *IEEE Open Journal of the Solid-State Circuits Society*, 2025.

[125] Harsh Sharma et al. HeMu: Energy-efficient DNN inferencing via heterogeneous-multi-chiplet architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[126] Alish Kanani et al. THERMOS: Thermally-aware multi-objective scheduling of AI workloads on heterogeneous multi-chiplet PIM architectures. *ACM Transactions on Embedded Computing Systems*, 24(5s):1–26, 2025.

[127] Y Kagawa and H Iwamoto. 3d integration technologies for the stacked cmos image sensors. In *2019 International 3D Systems Integration Conference (3DIC)*, pages 1–4. IEEE, 2019.

[128] Y Kagawa, H Hashiguchi, T Kamibayashi, M Haneda, N Fujii, S Furuse, T Hirano, and H Iwamoto. Impacts of misalignment on 1μm pitch cu-cu hybrid bonding. In *2020 IEEE International Interconnect Technology Conference (IITC)*, pages 148–150. IEEE, 2020.

[129] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 10–14, 2014.

[130] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551:232–236, 2017.