UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DOCTORADO EN CIENCIAS DE LA INGENIERÍA CON MENCIÓN EN INGENIERÍA ELÉCTRICA

Vision circuits architecture with on-chip spatial window operations

Profesor supervisor: PhD. Miguel Figueroa T.
Departamento de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de Concepción, Chile
Profesor co-supervisor: PhD. Payman Zarkesh-Ha
Department of Electrical and Computer Engineering
University of New Mexico, USA

Tesis presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al grado académico de Doctor en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica

BÁRBARO MAYKEL LÓPEZ-PORTILLA VIGIL
CONCEPCIÓN - CHILE
2024

Contents

List of figures

List of tables

List of acronyms

ADC Analog to Digital Converter

APS Active Pixel Sensor

ASIC Application Specific Integrated Circuit

CCD Charge Coupled Device

CPU Central Processing Unit

CIS CMOS Image Sensor

CTIA Capacitive Transimpedance Amplifier

CMOS Complementary Metal Oxide Semiconductor

DR Dynamic Range

DPS Digital Pixel Sensor

DRC Design Rule Checking

DSP Digital Signal Processing

EDA Electronic Design Automation

ERC Electrical Rule Checking

FN False Negatives

FP False Positives

FPGA Field Programmable Gate Arrays

fps frames per second

IEF Image Enhancement Factor

IoT Internet of the Things

LBP Local Binary Pattern

LoG Laplacian of the Gaussian

MIM Metal Insulator Metal

MSE Mean Square Error

M-IoT Multimedia Internet of Things

MAC Multiply-Accumulate

NCC Network Consistency Checking

NUC Non-uniformity Correction

OTA Operational Transconductance Amplifier

PDK Process Design Kit

PPS Passive Pixel Sensor

PFoM Pratt Figure of Merit

PPV Positive Predictive Value

PSNR Peak Signal to Noise Ratio

PWM Pulse Width Modulation

TN True Negatives

TP True Positives

SIMD Single Instruction Multiple Data

SNR Signal to Noise Ratio

SoG Sobel of the Gaussian

SSIM Structural Similarity Index Measure

SIS Smart Image Sensor

Abstract

CMOS Image Sensors (CIS) are crucial components in multimedia applications for the Internet of the Things (IoT). They allow IoT devices to perceive, interpret, and respond to the physical world, improving efficiency, safety, and convenience across various industries. The images captured by these sensors are sent to an external digital device for processing. Additionally, the images produced by CMOS sensors may contain defective pixels due to noise, manufacturing errors, or device malfunction. These defects need to be detected and corrected close to the sensor before the images are processed, which is essential for ensuring the images are helpful for human users as well as for image-processing and machine-vision algorithms.

This thesis reports the design and evaluation of two Smart Image Sensor (SIS) of 128×128 pixels using a 0.35 μm CMOS process, one for kernel-based spatial filtering calculation and the other for detecting and correcting defective pixels. The SISs are based on custom smart pixels that can perform some image processing algorithms in the analog domain. The remaining processing, including machine-vision algorithms and machine-learning methods, is handled by an external digital device. The focus of this thesis does not extend to this latter aspect.

The first SIS enables spatial filtering using a single kernel, two different kernels on the same image, or two cascaded kernels. The circuit calculates the absolute value of each convolution on-pixel in parallel at each pixel using simple neighborhood arithmetic during photocurrent integration. This process is repeated when the filter requires more than one kernel. According to post-layout simulation, this SIS processes images at frame rates ranging from 743 to 752 frames per second (fps) and achieves a fill factor of 20.6 %. Our SIS outperforms state-of-the-art circuits in denoising, being the best result with a Mean Square Error (MSE) of 0.90, Peak Signal to Noise Ratio (PSNR) of 48.55 dB, and Structural Similarity

Index Measure (SSIM) of 0.99, configured as the mean filter. Additionally, it demonstrates comparable edge detection performance to state-of-the-art circuits. The best results were achieved with a MSE of 4.68, PSNR of 41.42 dB, and SSIM of 0.97, using the y-direction Robert filter. Our proposed circuit demonstrated good edge-location accuracy of 0.99 % with the Sobel filter, which is one of the best results according to the Pratt Figure of Merit.

During photocurrent integration, the second SIS detects defective pixels in parallel at each pixel using simple arithmetic operations within a neighborhood. At the column level, the circuit replaces the defective pixels with the median value of their neighborhood. According to post-layout simulation, this SIS processes images at 694 fps, achieving a fill factor of 35.4 %. Our SIS produces better results than current state-of-the-art algorithms: it achieves a PSNR and Image Enhancement Factor (IEF) of 45 dB and 198.4, respectively, in images with 0.5 % random defective pixels, and a PSNR of 44.4 dB and IEF of 194.2, respectively, in images with 1.0 % random defective pixels.

The results demonstrate that, in a vision system, transferring part of the computation to the CIS can be beneficial. In this setup, an analog circuit performs computations at the pixel level using arithmetic operations within a pixel window while integrating photodiode current. This approach leverages the parallel architecture of the pixel array, enabling high frame rates, low power consumption, and compact size. Additionally, it minimizes the impact on fill factor and data accuracy.

Thanks to the results obtained in this thesis, our contributions are that a SIS is developed with on-pixel computing capabilities to perform kernel-based spatial filtering, allowing external configuration for various kernels and executing computations during photocurrent integration, achieving performance comparable to other SISs. Additionally, an algorithm from the literature for detecting defective pixels is modified for analog hardware implementation, yielding superior results compared to existing methods. Furthermore, another SIS is developed to detect defective pixels at the pixel level during photocurrent integration, implementing correction at the column level. This SIS is the first instance reported in the state of the art that combines detection and correction processes in the analog domain using CMOS technology on the same chip, achieving better outcomes than other implementations.

Resumen

Los sensores de imagen CMOS (CIS) son componentes cruciales en las aplicaciones multimedia del Internet de las cosas (IoT). Permiten a los dispositivos IoT percibir, interpretar y responder al mundo físico, mejorando la eficiencia, la seguridad y la comodidad en diversos sectores. Las imágenes captadas por estos sensores se envían a un dispositivo digital externo para su procesamiento. Además, las imágenes producidas por los sensores CMOS pueden contener píxeles defectuosos debido a ruido, errores de fabricación o mal funcionamiento del dispositivo. Estos defectos deben detectarse y corregirse cerca del sensor antes de procesar las imágenes, lo que es esencial para garantizar que las imágenes sean útiles tanto para los usuarios humanos como para los algoritmos de procesamiento de imágenes y visión artificial.

Esta tesis reporta el diseño y evaluación de dos sensores de imagen inteligentes (SIS) de 128×128 píxeles utilizando un proceso CMOS de 0.35 μm, uno para el cálculo de filtrado espacial basado en kernel y el otro para la detección y corrección de píxeles defectuosos. Los SIS se basan en píxeles inteligentes personalizados que pueden realizar algunos algoritmos de procesamiento de imágenes en el dominio analógico. El resto del procesamiento, incluidos los algoritmos de visión artificial y los métodos de aprendizaje automático, es gestionado por un dispositivo digital externo. El enfoque de esta tesis no se extiende a este último aspecto.

El primer SIS permite el filtrado espacial utilizando un único kernel, dos kernels diferentes en la misma imagen o dos kernels en cascada. El circuito calcula el valor absoluto de cada convolución en paralelo en cada píxel utilizando simple aritmética en la vecindad durante la integración de la fotocorriente. Este proceso se repite cuando el filtro requiere más de un kernel. Según la simulación post-layout, este SIS procesa imágenes a frecuencias de cuadro que varían entre 743 y 752 fotogramas por segundo (fps) y alcanza un fill factor de 20.6 %. Nuestro

SIS supera a los circuitos del estado del arte en la eliminación de ruido, siendo el mejor resultado con un Error Cuadrático Medio (MSE) de 0.90, una Relación Peak Señal a Ruido (PSNR) de 48.55 dB, y una Medida del Índice de Similitud Estructural (SSIM) de 0.99, configurado como el filtro mean. También demuestra un rendimiento en la detección de bordes comparable al de los circuitos del estado del arte. Los mejores resultados se obtuvieron con un MSE de 4.68, PSNR de 41.42 dB, y SSIM de 0.97, utilizando el filtro Robert de dirección y. Nuestro circuito propuesto demostró una buena exactitud de localización de bordes del 0.99 % con el filtro Sobel, que es uno de los mejores resultados según la figura de mérito de Pratt.

Durante la integración de la fotocorriente, el segundo SIS detecta los píxeles defectuosos de forma paralela en cada píxel utilizando operaciones aritméticas simples dentro de una vecindad. A nivel de columna, el circuito sustituye los píxeles defectuosos por el valor medio de su vecindario. Según la simulación post-layout, este SIS procesa las imágenes a 694 fps, alcanzando un fill factor del 35.4 %. Nuestro SIS muestra mejores resultados que los algoritmos del estado del arte, alcanzando una PSNR y un factor de mejora de la imagen (IEF) de 45 dB y 198.4, respectivamente, en imágenes con un 0.5 % de píxeles defectuosos aleatorios, y una PSNR de 44.4 dB y un IEF de 194.2, respectivamente, en imágenes con un 1.0 % de píxeles defectuosos aleatorios.

Los resultados demuestran que, en un sistema de visión, transferir parte del cálculo al CIS puede ser beneficioso. En esta configuración, un circuito analógico realiza cálculos a nivel de píxel utilizando operaciones aritméticas dentro de una ventana de píxel mientras integra la corriente del fotodiodo. Este enfoque aprovecha la arquitectura paralela de la matriz de píxeles, lo que permite altas frecuencias de cuadro, un bajo consumo de energía y un tamaño compacto. Además, minimiza el impacto sobre el fill factor y la exactitud de los datos.

Gracias a los resultados obtenidos en esta tesis, nuestras aportaciones son que se desarrolla un SIS con capacidad de cómputo sobre el píxel para realizar filtrado espacial basado en kernels, permitiendo la configuración externa para varios kernels y ejecutando los cálculos durante la integración de la fotocorriente, consiguiendo un rendimiento comparable a otros SISs. Además, un algoritmo de la literatura para la detección de píxeles defectuosos se modifica para la implementación de hardware análogo, dando resultados superiores en comparación con los métodos existentes. Además, se desarrolla otro SIS para detectar píxeles defectuosos a nivel de píxel durante la integración de fotocorriente, implementando la corrección a nivel de columna. Este SIS es el primer caso reportado en el estado del arte que combina procesos de detección y corrección en el dominio análogo utilizando tecnología CMOS en el mismo chip, consiguiendo mejores resultados que otras implementaciones.

1 Introduction

1.1 General introduction

Internet of the Things (IoT) is rapidly transforming our world, with more than 75 billion smart devices expected to be connected by the end of 2025 [2]. The IoT is a network of devices embedded with sensors, software, and network connectivity to collect and share data without human intervention [3, 4]. Users can access and utilize the data outputs thanks to the integration of embedded devices within the internet infrastructure facilitated by the IoT. The IoT has experienced significant growth in recent years, driven by its widespread adoption in various consumer and industrial applications. These include smart homes, environmental monitoring, transportation systems, industrial connectivity, security, and professional services [2, 5].

The explosion of multimedia-on-demand traffic, referring to images and video between devices connected to each other and to the Internet, has drastically changed the vision of the IoT from scalar to Multimedia Internet of Things (M-IoT) [6, 7]. Although IoT devices are typically resource-constrained, with limited processing power, memory, and battery life, this is especially notable for M-IoT devices due to three factors: capturing images from an image sensor, transmitting images, and processing images on the host processor [6, 7].

A wide range of M-IoT applications capture images using a CMOS Image Sensor (CIS), which captures the image of its surroundings by converting incident light into proportional electrical signals. and perform image processing to extract meaningful information. For example, the OV6620 CIS protects against rice spoilage by capturing images of the Nilparvatha lugens insect in wireless sensor nodes [8], while the MT9V034 CIS acts as the eyes of the SmartVision system to ensure the safety of railroad bridges [9]. In industrial environments, the

OV2640 CIS automates and records measurements from circular gauge images in hazardous or hard-to-reach areas, ensuring precise and safe operation [10], and the AR0231AT CIS captures images that utilize driverless public transportation managed by passengers themselves in urban environments [11].

CIS are preferred over Charge Coupled Device (CCD) in portable devices based on IoT technologies due to their higher dynamic range and lower current and voltage requirements [12]. In addition, recent advances in deep submicron Complementary Metal Oxide Semiconductor (CMOS) have brought significant advantages to CIS, such as compatibility with standard CMOS technology, the ability to integrate sensors with analog and digital circuitry for pixel-level image processing, random access to image data, low power consumption, and high-speed image acquisition [13, 14, 15, 12, 16].

The CIS collects a substantial amount of environmental data, which is transmitted and subsequently processed by a digital processor—typically a Central Processing Unit (CPU) in the IoT server or an embedded co-processor in the IoT node. On-node processing provides faster data access, lower latency response, high security, and reduced communication bandwidth. However, the hardware above mentioned may need more performance for video processing or may be costly in terms of power, resources and size. These programmable solutions can be a limitation in applications that require on-node processing.

To address these challenges, dedicated digital hardware, such as Field Programmable Gate Arrays (FPGA) devices, can facilitate portable on-node processing [17, 18]. However, the serial data transmission from the CIS to digital hardware can negate the benefits of on-node processing and hinder the parallelization of algorithmic tasks. This creates a bottleneck that obstructs the real-time execution of image and video processing, as well as vision algorithms [19, 20, 21]. Furthermore, this sensor-processor approach is not well-integrated into IoT applications, as it consumes substantial power, which is unavailable in IoT nodes, and generates high latency during image data transmission, limiting frame rates and pixel resolution [22, 23, 24].

The sensor-processor problem is not new and has been addressed in the literature, with proposed solutions that involve offloading some processing tasks directly onto the CIS [25, 26, 27, 21], transforming it into a Smart Image Sensor (SIS). By integrating processing capabilities within the sensor itself, the SIS effectively alleviates the limitations of traditional sensor approach. This approach allows for a more efficient division of processing tasks between the sensor and external digital hardware. SIS research shows they can provide fast, simultaneous data acquisition and processing, low power requirements, ease of fabrication using digital CMOS technology, and affordability [16, 28, 21, 29, 30], making them attractive for resource-constrained M-IoT systems. The SIS implements image processing algorithms, usually preprocessing algorithms, on the same die to im-

prove image quality or extract some information. These algorithms are typically implemented in the analog domain within the CIS, making them smaller and more power efficient than their digital counterparts [31, 32]. SIS takes advantage of the CMOS-based technology of the CIS to add this circuitry [33, 21].

Several SISs can be found in the literature implementing spatial filters, such as kernel-based spatial filters for noise reduction [34, 35, 36] and edge detection [37, 19, 38]. Although these SISs can be part of an IoT network, the processes of acquiring the photodiode value and calculating the convolution are in different phases, resulting in decreased frame rates per second, such as 480 frames per second (fps) [39], 100 fps [40], and 60 fps [33], all at resolution similar to ours proposed CISs. It causes delays between capturing and reacting to a scene in critical applications such as robotics or autonomous vehicles.

According to the literature, there is significant interest in the design of SIS circuits and their application in various fields, including IoT. While many SIS have already been developed, the growing use of portable devices, multimedia systems, and information acquisition technologies in our daily lives demands increasingly strict characteristics, such as low power consumption, compact size, high resolution, high accuracy data and high frame rates. In this context, we focus on two types of processing: noise reduction and edge detection and the detection and correction of defective pixels. These processes enhance image quality and enable the extraction of relevant information for both human users and image-processing or machine-vision-based systems.

This thesis presents the design and evaluation of two new 128×128 pixel SIS circuits that can compute two image preprocessing algorithms: 3×3 pixel kernel-based spatial filtering and detecting and correcting defective pixels. Spatial filtering is performed at the pixel level to enhance image quality (denoising) and extract features by edge detection. Defective pixel detection is also employed to enhance image quality and is conducted at the pixel level. Similarly, pixel correction is performed for the same objective but at the column level to avoid a reduction in the photodetector's effective area. Both SISs comprise Capacitive Transimpedance Amplifier (CTIA)-based smart pixels that perform computations of the algorithms during photocurrent integration. This technique not only increases the frame rate per second but also reduces the impact of device mismatch and nonlinearity on the accuracy of the results while lowering the area overhead of the computation circuits. Both SISs are suitable for integration into portable devices, where they can be used to acquire images of a scene or, depending on their architectures, one of the two types of pre-processing mentioned above.

1.2 Hypothesis

The thesis hypothesizes that integrating analog circuits into a SIS can enhance image preprocessing by performing computations directly within the pixel. By utilizing window operations and leveraging the parallel architecture of the pixel array during the photodetector current integration process, this approach aims to meet the requirements for high frame rates, compact size and high accuracy, while minimizing the impact on the fill factor. These characteristics are essential for integrating the proposed SIS into the portable devices typical of modern IoT systems.

1.3 General objective

The general objective of this thesis is to design and evaluate a Smart Image Sensor that employs analog integrated circuit design techniques with CMOS technology to perform image preprocessing algorithms directly on the pixel, thereby achieving optimal frame rates, compact size, high accuracy, and minimal impact on the fill factor.

1.4 Specific objectives

1. Define digital image processing algorithms in the literature to identify an algorithm based on pixel-level operations that can be parallelized.

2. Adapt the algorithms to the specific hardware constraints within each sensor pixel.

3. Design the analog circuits using CMOS technology to implement the algorithms, integrating them within each smart sensor pixel of the SIS.

4. Evaluate the performance of the SIS using standard image quality and sensor-specific metrics.

1.5 Scope

The scope of this thesis includes several key areas: the research and proposal of SIS circuits, as well as the simulation and evaluation of their performance in terms of speed, area, and accuracy. The results of this work will be prepared for publication in journals indexed by Web of Science. The two SISs are designed on a CMOS technology process, but future adaptations using alternative technological

processes will also be considered. This research will exclusively focus on the SIS circuit, omitting the Analog to Digital Converter (ADC) design and any computations performed on external digital devices. According to the general objective of designing a SIS, the algorithms evaluated and reviewed in this thesis will be restricted to low-complexity ones found in the literature. Consequently, the literature analysis will focus on studies relevant to the two implemented applications: kernel-based spatial filtering for denoising and edge detection and defective pixel detection and correction.

1.6 Contribution

The major contributions of this research thesis are highlighted as listed:

1. A SIS is developed with computing capability on-pixel to perform kernel-based spatial filtering. This imager can be externally configured to operate with different kernels, and the computation is performed during photocurrent integration. Its performance is comparable to state-of-the-art SISs.

2. The algorithm presented in [41] for detecting defective pixels is modified and adapted for its implementation in analog hardware. The resulting algorithm shows better results than those existing in the state-of-the-art.

3. A SIS is developed with computing capability to detect defective pixels at the pixel level, with correction implemented at the column level. The computation is performed during the integration of the photocurrent. This circuit is the first reported in the state-of-the-art to perform detection and correction processes in the analog domain with CMOS technology on the same chip and to report better results than other state-of-the-art implementations.

1.7 Thesis structure

The rest of this thesis report is organized as follows:

Chapter 2 discusses the literature review. It overviews imaging systems, CIS, and SIS. In addition, a literature review is presented that focuses on two types of image preprocessing to be implemented at the pixel level in the CIS: kernel-based spatial filters for denoising and edge detection, and techniques for detecting and correcting defective pixels. Chapter 3 presents the design and evaluation of the SIS, which computes a kernel-based spatial filtering image at the pixel level with the proposed denoising and feature extraction by edge detection. Chapter 4 presents the design and evaluation of the SIS that computes the detection and

correction of defective pixels at the pixel and column level, respectively. Chapter 5 concludes the whole thesis and discusses some interesting future research directions.

2 Literature review

2.1 Introduction

This chapter introduces the fundamentals of CMOS Image Sensor (CIS) and their evolution into Smart Image Sensor (SIS). It also provides a literature review of existing Smart Image Sensors, focusing on two key applications: kernel-based spatial filtering and detecting and correcting defective pixels. Finally, it discusses the issues addressed, reinforcing the motivation behind this thesis.

2.2 Fundamentals of CIS

2.2.1 Imaging system

A computer vision system enables machines to interpret and understand visual information from the world around them, mimicking human vision to perceive, analyze, and make decisions based on digital images or videos [16, 24]. By leveraging techniques such as image processing, pattern recognition, and machine learning algorithms, computer vision systems can detect objects, classify scenes, recognize faces, and even estimate spatial relationships within an environment.

More advanced systems use dedicated hardware to accelerate the processing stage and gain portability, including microcontrollers, Application Specific Integrated Circuit (ASIC)s, or Field Programmable Gate Arrays (FPGA)s. However, conventional vision system architectures typically separate the information (image) capture and processing modules. Image sensors capture the information, converting the incident light into electrical signals using photosensitive elements [42]. These devices achieve performances similar to those of the human eye, and their core operation consists of two steps: first, the image sensor converts the incident photons into electron–hole pairs, and then, it transforms the resulting current into

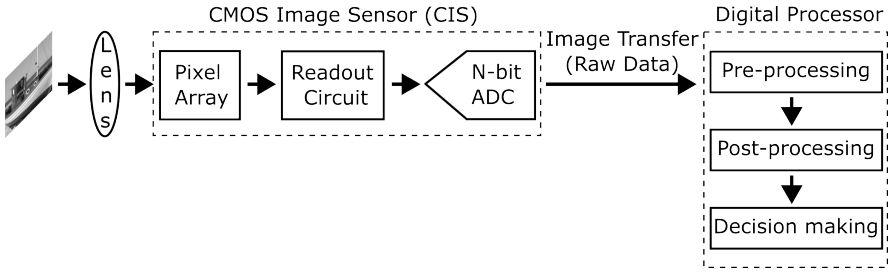


Figure 2.1: Conventional vision system architecture.

a voltage [43, 44]. The voltage at each photosensitive element is proportional to the number of photons received during the acquisition time. Currently, Charge Coupled Device (CCD) and CIS are the preferred technologies for digital imaging devices [45, 28, 16]. Although CCDs often deliver better image quality and low-light robustness, CIS are preferred on mobile applications due to their higher dynamic range and smaller current and voltage requirements. Furthermore, recent advances in deep submicron Complementary Metal Oxide Semiconductor (CMOS) have brought significant advantages for CIS, such as compatibility with standard CMOS technology, the possibility of integrating sensors with analog and digital circuits for image processing at the pixel level, random access to image data, low power consumption, and high-speed image acquisition.

A CIS-based vision system is shown in Figure 2.1, which consists of an optical lens, a CMOS image sensor, and a digital processor. The scene's light is directed through a lens onto a sensor plane containing a two-dimensional array of pixels. Each pixel utilizes a photodiode to convert incoming light into a voltage signal. This voltage signal undergoes conditioning prior to entering an Analog to Digital Converter (ADC). The resulting digital signal from the ADC accurately represents the illumination intensity at each pixel. Subsequently, these digital signals are transmitted to a digital processor as an image for processing and decision-making purposes.

2.2.2 Image sensor architectures

The most important component of an image sensor is its two-dimensional array of pixels that detect light rays, as shown in Figure 2.2. Each pixel contains a photodetector that measures light intensity, shown in gray. The remaining area of the pixel, shown in white, consists of a transistor-based circuit that is essential for processing the signal from the photodetector.

During readout, the row select activates a specific row, allowing its pixels to connect to readout circuits that convert the captured light into voltage signals. These signals are then sent off the sensor chip. The readout block comprises

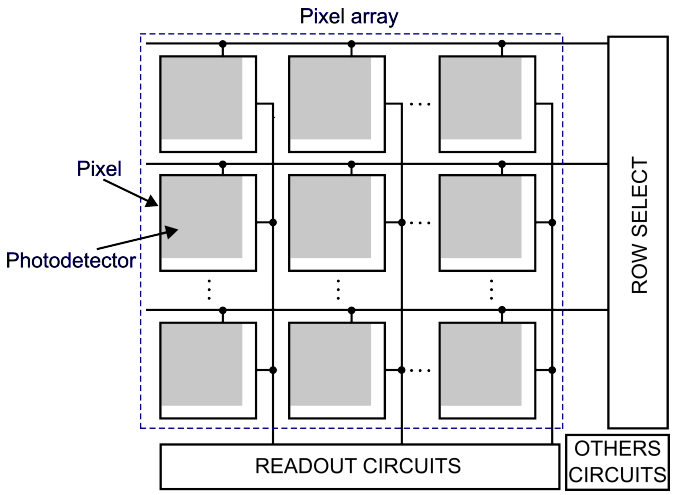


Figure 2.2: General CIS architecture.

signal conditioning circuits such as amplification, row-noise correction, and black-sun correction, along with ADCs. Additionally, it may include timing circuits, bandgap and reference generation circuits, configuration registers, and other related components.

The position of the ADC within a sensor significantly impacts system performance in terms of speed, power, and area. Based on its location, CIS architectures can be categorized into three main types: chip-wise, column-wise, and pixel-wise, illustrated in Figure 2.3. In the chip-wise architecture (Figure 2.3a), all pixel signals are sequentially read and converted by a single ADC. An analog buffer selects column outputs for conversion, making this architecture compact with the smallest footprint. However, its processing speed is often constrained, particularly for real-time and high-resolution imaging applications.

Conversely, the column-wise architecture (Figure 2.3b) enhances processing speed by employing multiple ADCs dedicated to a column. This approach reduces the speed and power demands on the buffer and ADC compared to chip-wise designs. Nonetheless, column-parallel ADC introduces challenges like increased area overhead and column-to-column variation, necessitating additional circuitry for mitigation.

An alternative for even greater imaging speed is the pixel-wise architecture (Figure 2.3c), where each pixel integrates its own ADC, allowing simultaneous digitalization of all pixels. Although this configuration achieves the highest speed, it enlarges pixel size, increasing sensor area and impacting power consumption due to greater parasitic capacitance and circuit leakage. Moreover, the limited space within each pixel restricts the use of additional circuit techniques, potentially constraining ADC performance.

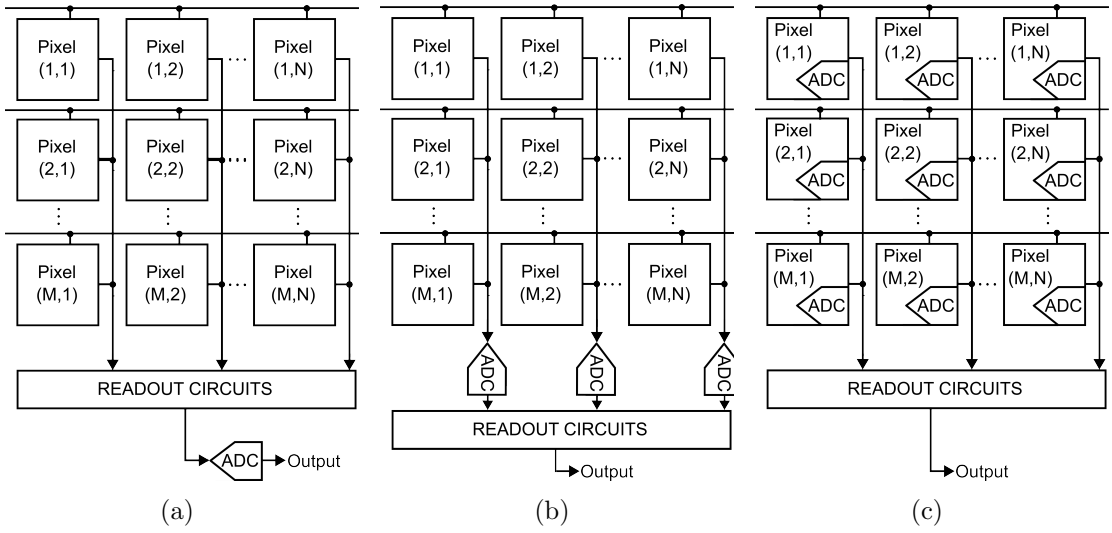


Figure 2.3: CIS architectures: (**a**) Chip-wise, (**b**) Column-wise, and (**c**) Pixel-wise.

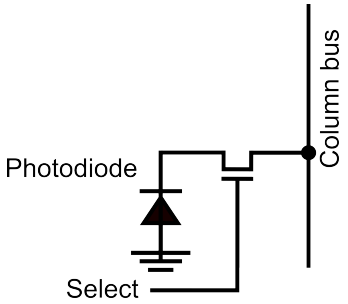


Figure 2.4: PPS architecture.

2.2.3 Pixel architectures

A pixel is the fundamental unit of a CIS, and its architectural configuration is consistent across all pixels within the CIS. Pixel architectures are classified into active or passive pixels. The distinction between the two types of pixels depends on whether the pixel contains signal amplification circuitry.

The Passive Pixel Sensor (PPS) consists of a photodiode and a switching transistor connected to the column readout bus [46, 47], as shown in Figure 2.4. The photodetector predominantly occupies the area of the PPS pixel. However, due to the significant column parasitic capacitance, conventional PPS image sensors are susceptible to KTC noise with a low Signal to Noise Ratio (SNR).

In contrast, Active Pixel Sensor (APS) incorporates a local active buffer, typically a source follower transistor, in conjunction with the reset and selection

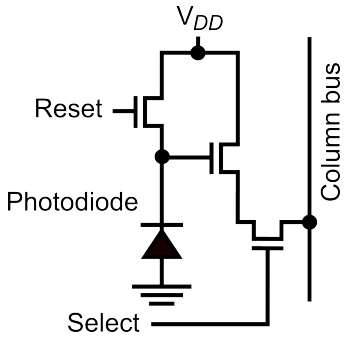


Figure 2.5: 3T-APS architecture.

transistors [46, 48]. This configuration results in a lower area for the photodetector than for passive pixels. Nevertheless, APS topologies also offer advantages. For example, the reduction in capacitance within each pixel decreases in read noise for the array, thereby increasing the Dynamic Range (DR) and SNR. Figure 2.5 shows a standard 3T-APS architecture. Some active-pixel architectures more common are 4T-APS [49], Digital Pixel Sensor (DPS) [50], Current Mode Pixel [51], and Capacitive Transimpedance Amplifier (CTIA) Pixel [29].

The essential element within both types of pixels is the photodetector. The most commonly employed version is the photodiode due to its straightforward integration into a standard CMOS process. Any parasitic PN junction originating from the N-Well or P-Well can be used as a photodetector.

2.3 From CIS to SIS

As illustrated in Figure 2.1, the CIS sequentially transmits the complete frames, pixel by pixel, to the digital processor for subsequent processing. This approach has significant drawbacks, including the inability to parallelize algorithmic data, increased latency and power consumption, limited frame rate, and substantial memory and computational demands on the digital processor [28, 19, 26]. Let us consider a CMOS image sensor with a typical resolution of 500×500 pixels [52, 53, 54] and a 10-bit ADC circuit. In this case, the amount of data per frame is 312.5 kbytes. At a standard video frame rate of 30 frames per second (fps), a minimum required for many consumer and professional devices, the total data transmission rate requires 9.4 Mbytes per second. This substantial data volume leads to challenges such as increased latency, network bandwidth constraints, higher storage costs, and the need for extensive hardware resources for data conversion, processing, and transmission [28, 16, 55].

A novel approach has been devised to address the limitations above, drawing inspiration from biological organisms that achieve high efficiency and perfor-

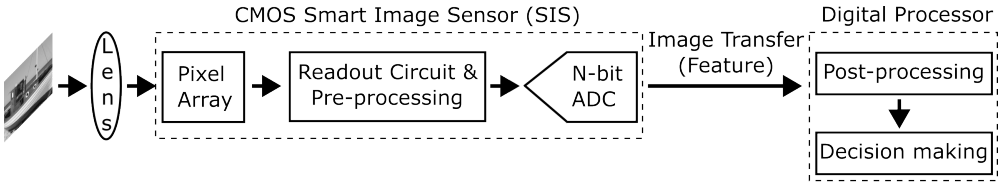


Figure 2.6: Modified vision system architecture.

mance by first operating locally on the data at the image sensor in parallel and then transferring the resulting information to a central processing system that performs higher cognitive functions. In this approach, a portion of the processing originally conducted by the digital processor is transferred to the CIS embedded within the pixel. The algorithms implemented in the CIS are typically designed to enhance image quality and facilitate subsequent analysis, a process known as preprocessing. These algorithms must process a substantial amount of data and are highly demanding regarding power and speed. In contrast, while the amount of data may be smaller in post-processing, the complexity of the algorithms is often greater. Incorporating computational capability at the pixel level enables the CIS to become a SIS, thereby establishing a novel vision system architecture, as shown in Figure 2.6. This transformation allows the SIS to acquire images and process them to extract features, thereby eliminating redundant information and reducing the data throughput for the digital processor [16, 28, 21, 29, 30].

However, incorporating computation into the pixel presents some challenges. First, adding circuitry reduces the area available to the photodetector, which can result in less light being captured by the sensor. In addition, the computational circuitry is often analog, which can decrease the accuracy of the data. Despite these disadvantages, carefully choosing the algorithm to implement in the pixel and designing the circuit with the minimum number of transistors and correct dimensions can result in a SIS with good performance.

Research on SIS using the previous approach shows that they can provide fast and simultaneous data acquisition and processing, low power requirements, and ease of fabrication using digital CMOS technology [16, 28, 21, 29, 30], making them attractive for resource-constrained Internet of the Things (IoT) systems.

This thesis is focused on two types of image preprocessing to be implemented at the pixel level in the CIS. Section 2.4 reviews the existing literature on the kernel-based spatial filters for denoising and edge detection in preprocessing. Section 2.5 reviews the existing literature on the techniques for the detection and correction of defective pixels.

2.4 Kernel-based spatial filtering image sensors

In image and video processing systems, preprocessing is crucial for enhancing overall quality and making subsequent analysis more effective. One common preprocessing technique is the use of two-dimensional spatial filters, which significantly impact subsequent processing stages and overall system performance. However, implementing these filters in portable hardware devices presents additional challenges due to constraints such as power consumption, size limitations, and resource utilization [56, 45, 57]. Digital Signal Processing (DSP)s, ASICs, and FPGAs are specialized chips commonly used for portability, as evidenced by numerous studies [58, 59, 60].

FPGAs are the most attractive portable solution for implementing image and video processing algorithms among these platforms. These devices can exploit fine-grained parallelism, enable reusable computation, and implement pipelined architectures, providing an excellent performance/area tradeoff and high flexibility [61, 62, 63]. These characteristics allow for the implementation of efficient kernel convolution-based image processing algorithms [64, 65]. Furthermore, there are specific filter designs based on kernel properties, such as symmetry [66] or separability [67], that can be implemented on these platforms to improve performance metrics.

While digital hardware such as FPGAs offers excellent capabilities for image and video processing, the serial data transfer from CIS can limit parallelism, introduce latency, and increase power consumption [21, 29, 30]. In contrast, SIS provides an alternative to this drawback by utilizing an in-sensor approach that combines CIS and analog processing with a fine-grained architecture. Additionally, this design strategy reduces memory requirements, latency, and power consumption, making it optimal for resource-constrained Multimedia Internet of Things (M-IoT) applications. This approach compromises some computation accuracy and reduces the photodiode area [31, 32, 68, 69]. Some works proposed a SIS that performs edge detection or noise removal using a specific kernel [70, 71, 72, 38, 73, 74, 75, 34, 35]. These architectures have the advantage of low power and resource consumption and typically produce high frame rates per second. In [70], the authors implemented analog edge detection based on a variation of the Robert operator. The computation is performed by a cascaded bump circuit integrated directly next to a photodiode array. The results showed improvement over [71] in detection sensitivity, frames per second, and power consumption. Garcia-Lamont [76] presented an analog SIS architecture to compute the Prewitt edge detection filter using a simple analog arithmetic circuit integrated with each photodiode in the imaging array. In [72, 38], the authors proposed a SIS that uses a simple kernel, which requires only a single operation to detect edge data. This approach leverages mature APS architecture and single-slope

ADCs and incorporates a minimal digital circuits, leading to low power consumption and processing time. The authors of [73, 74, 75] proposed a retinal circuit inspired by biology that utilizes only current mirrors and current subtractors for the convolution operation, with all processing in the current domain. In [73], the authors used a custom kernel, while in [74] they used the Laplacian kernel. In [75], they used a high-pass filter kernel. The objective is to enhance the patient's visual perception. In the first two cases, they counteract the effects of the lateral inhibition phenomenon in the human retina, while in the last case, they increase contrast. Soell et al. [77] proposed an intelligent analog image sensor that can perform the Sobel algorithm, calculate absolute values in both the x and y directions, and sum using operational amplifier-based circuits. Finally, it can output a 1-bit image using a comparator. Shi et al. [34] reported a SIS that can act as a mean or binomial spatial filter to reduce image noise. Irmanova et al. [35] presented a method for identifying similarities between a central pixel and its surrounding pixels within a given filter window using on-sensor neuromorphic vision hardware. These similarity scores are then used to construct an adaptive denoising filter. Unlike [34], which only used circuits based on classical configurations of operational amplifiers, [35] included comparators and digital memory circuits.

Other SIS architectures in the literature used kernels that allow value modification and offer more versatility, but they increase die area and power consumption [78, 79, 80, 33]. Dubois et al. [78] proposed kernel-based spatial gradient computation. The pixel-level processing element performs a linear combination of the four adjacent pixels using a four-quadrant multiplier and adder circuit with the four associated weights. Hsu et al. [79] presented a programmable 3×3 kernel for convolution-based feature extraction. Besides, the imager features the Pulse Width Modulation (PWM) pixel and eight-directional analog Multiply-Accumulate (MAC) operation of 3×3 subarray. Chen et al. [80] reported a kernel-readout CIS and analog computation close to the sensor unit. This enables the MAC operation, pooling, and nonlinear function operations for convolutional computation. Song et al. [33] proposed a processing-in-pixel architecture that performs convolution by modulating the exposure time of photodiodes in each pixel. The resulting values are stored on capacitors, and charge redistribution seamlessly integrates the multiplication results and performs the accumulation.

In summary, previous research indicates an interest in using spatial filters for image and video processing on hardware, specifically for edge detection and noise reduction. These tasks are essential to enhance image quality and enable extracting relevant information for human users and image-processing or machine-vision-based systems. The focus is on hardware implementations that utilize the low power and parallel processing capabilities of CMOS technology, making it suitable for IoT devices. Performing computations directly at the pixel level offers

a feasible alternative to traditional digital processors. This approach achieves low power consumption and high frame rates required by portable devices while reducing the impact of device mismatch and nonlinearities on accuracy. In-pixel analog processing during photocurrent integration reduces circuit area overhead with minimal cost of accuracy.

2.5 Detection and correction defective pixels image sensors

The defective pixel rate is one of the most significant characteristics used to evaluate the quality of an image sensor at the end of its manufacturing process and during its useful life [81, 82, 83]. The occurrence of defective pixels results in degradation of the overall visual quality of the captured image, which can have a critical effect on its subsequent interpretation, mainly if additional processing is performed on the image. For example, using CIS in IoT systems is very attractive for autonomous vehicles. Continuous data collection in the form of images enables real-time analysis of the vehicle's environment, facilitating the decision-making necessary for proper operation. However, defective pixels can compromise this capability, leading to erroneous decisions or inaction, putting the lives of passengers at risk and causing damage to the vehicle [84]. On the other hand, in home automation, IoT systems use cameras with CIS for biometric recognition and home security. However, defective pixels in these cameras can cause errors in person identification or motion detection, compromising system efficiency and providing an unsafe environment for people [85]. Therefore, defective pixel detection and correction algorithms are crucial for correctly interpreting the information in the image. In this work, we focus on online algorithms, which detect defective pixels in the image sensor from one or more video frames acquired during normal operation in real time [83, 86, 87]. In particular, we focus on single-frame defective pixel detection algorithms, which are faster and require less memory than multi-frame algorithms.

Chan [88] proposed a dead pixel detection method that labels a pixel as defective when its value lies outside of an interval defined as $[a - r, a + r]$, where a is the average of its eight neighbors except for the second-to-largest and second-to-smallest values and r is the range between these two values. Cho et al. [89] detected defective pixels based on whether the difference between its value and the average between two of its neighbors is greater than the mean of the pixel and its four neighbors or the maximum pixel value minus this mean. These algorithms use only basic operations such as absolute difference, comparison, spatial neighborhood averages, and rank-ordering. Although they can be implemented efficiently, these algorithms produce high false positive rates, increasing the number

of pixel corrections and degrading the resulting image quality [41]. El-Yamany [90] showed a robust method of identifying singlets and couplets of defective pixels that uses comparison, averaging, minimum, and maximum operations. They used two tunable parameters that control the detection sensitivity and specificity. Tchendjou et al. [41] proposed an algorithm that flags defective pixels when the difference between the actual and estimated value is greater than a threshold (or the maximum pixel value minus this threshold). The threshold is computed as the mean between the pixel value and a simple average of its eight neighbors. The estimated value is computed as a weighted (or, in the simplified version, unweighted) average of the adjacent pixels. They achieved better sensibility, specificity, Positive Predictive Value (PPV), and phi-coefficient than [88, 89]. The authors of [91] proposed an optimization algorithm to detect and correct defective pixels in an image using two defective pixel detection methods; the first uses the distance between the pixel and its neighbors [41], while the second uses the dispersion of the pixel values within a neighborhood window [92]. Yongji et al. [93] used a method similar to Chan [88] for defective pixel detection, except that they used a 5×5-pixel window to compute the average and excluded the two largest and the two smallest pixels in this window.

The algorithms described above can achieve good accuracy; however, they must be programmed on high-performance processors in order to achieve short processing times. While the subsequent cost in die area and power consumption of these solutions may not be a limitation in larger applications, they are typically unsuitable for portable and mobile devices. Consequently, many researchers have proposed special-purpose processing systems based on custom hardware architectures in order to focus on speed, portability, and low power consumption [94, 86, 92, 83]. In fully-digital solutions, FPGA are a widely used platform to implement these architectures thanks to their large fine-grained parallelism and relatively low cost and power consumption compared to traditional programmable processors. Kumar et al. [94] implemented real-time non-uniformity correction and defective pixel detection on a Xilinx XC2S1500 FPGA using a thresholding criterion based on the mean $\pm$ 3 times the standard deviation of the pixel values within a neighborhood. In [86], the authors implemented a combination of offline and online defective pixel detection and correction on a Lattice Semiconductors ECP2 FPGA. In each frame, they computed the difference between the current pixel and the average of its neighbors and then compared it to a threshold in order to select candidate defective pixels. If the same pixel was detected in consecutive frames, that pixel was considered to be defective. In [92], the authors used the median, standard deviation, range, interquartile range, and other statistical dispersion parameters of pixel blocks to detect and correct defective pixels in an image. Running their algorithm on a Microblaze microcontroller embedded in a Xilinx VC707 FPGA, the quality of their results was only slightly lower than those

in [41] while achieving a 3.5 times faster computation time. Tchendjou et al. [83] presented a heterogeneous architecture on a Xilinx ZC702 device implementing three detection algorithms: the first was the algorithm presented in [41]; the second was similar to the first, except that it computed the estimated pixel value as the median of the neighborhood window; the third, described in [92], was faster and able to achieve slightly better results than the first two, though it used more hardware resources. Moreover, the last algorithm required five parameters to be determined empirically based on the images in the dataset.

Despite the advantages of the architectures described above, these custom digital processors operate outside the CIS, accessing the pixel values sequentially after analog to digital conversion. This limits the intrinsic data parallelism of the algorithm, reducing the throughput and increasing the implementation latency. Moreover, sequential access to the pixel values requires line buffers in the digital processor to implement window-based operations. As an alternative, all or part of the operations of the algorithm can be performed in a SIS at the pixel or column level, usually via analog circuits integrated into the image sensor [95, 96, 40]. This approach greatly increases the parallelism of the implementation, though at the cost of increasing the pixel size or decreasing the area available for photodetectors. Ginhac et al. [97] reported a dynamically configurable parallel processor array integrated into a CMOS image sensor. A Single Instruction Multiple Data (SIMD) architecture enabled programmable kernel-based image processing computation on each pixel. Garcia-Lamont [76] described an analog SIS to compute the Prewitt edge detection algorithm using a simple arithmetic analog circuit integrated with each photodetector in the imaging array. Suárez et al. [98] presented a SIS for Gaussian pyramid extraction with per-pixel analog processing circuits and local analog memories. Gottardi et al. [99] proposed a vision sensor that computes a four-bit Local Binary Pattern (LBP) for each pixel during the integration time. They demonstrated using these LBPs to perform software-based image description and retrieval.

The research described above shows the importance of detecting and correcting defective pixels before displaying or processing an image. While most hardware implementations of these algorithms have used digital programmable logic devices, research shows that implementing the computation at the pixel or column level can greatly improve the parallelism of the implementation. Moreover, performing the computation in the analog domain during photocurrent integration can reduce the impact of device mismatch and nonlinearity on the accuracy of the results and lower the area overhead of the computation circuits.

2.6 General discussion

High-performance, fully digital solutions, such as FPGA, are extensively employed in computing tasks involving substantial data volumes and high computational complexity, particularly in digital image processing, computer vision, and machine learning. However, these solutions typically receive data serially, necessitating line buffering and conducting processing post-acquisition and digitization of image data. This sequential approach increases communication bandwidth requirements and power consumption. Research indicates that integrating SISs with fully digital devices presents a robust alternative to enhancing overall system performance, notably in contemporary IoT systems.

A literature review reveals a critical and growing interest in applying spatial filters for image and video processing on hardware, with a significant focus on edge detection and noise reduction. Additionally, addressing defective pixels before an image is displayed or processed is essential for maintaining the accuracy and reliability of image processing systems. These issues are not just peripheral but central to ensuring the system's overall performance. To this end, our research emphasizes developing hardware implementations that leverage the advantages of CMOS technology, including its low power consumption, compact size, high frame rate capability, and parallel processing efficiency. These attributes make CMOS technology a viable solution for enhancing IoT devices, tackling these critical challenges directly, and setting a new standard for performance and reliability in image processing.

The enhancement of image quality through the reduction of noise and the correction of defective pixels is of paramount importance in achieving reliable post-processing outcomes. The process of edge detection is essential for the extraction of image features and can reduce the data volume transmitted to high-performance digital devices. In all cases, the level of computing is low, focusing primarily on pixel-wise addition and multiplication operations within spatial image windows. This operational simplicity facilitates efficient implementation in analog hardware, as analog circuits can manage basic pixel-wise computations effectively with less complexity than digital systems.

Both kernel-based spatial filtering and defective pixel detection benefit from direct pixel-level computations, offering a viable alternative to conventional digital processors. This approach capitalizes on SISs' exploitation of pixel-level parallelism, achieving high frame rates and low power consumption, which are crucial for portable devices. Moreover, conducting computations in the analog domain during photocurrent integration mitigates accuracy issues from device mismatches and nonlinearities. In-pixel analog processing during photocurrent integration also curtails circuit area overhead with minimal compromise on accuracy.

This review demonstrates the potential of SISs in revolutionizing image pro-

cessing paradigms and there is significant interest in the design of these circuits and their application in various fields, including IoT. While many SIS have already been developed, the growing use of portable devices, multimedia systems, and information acquisition technologies in our daily lives demands increasingly strict characteristics, such as low power consumption, compact size, high resolution, and high frame rates. In this context, we focus on two types of processing: noise reduction and edge detection and the detection and correction of defective pixels. These processes enhance image quality and enable the extraction of relevant information for both human users and image-processing or machine-vision-based systems.

3

On-chip kernel-based spatial filter for CMOS Image Sensor

3.1 Introduction

This chapter outlines the proposed method for kernel-based spatial filtering using convolution, focusing on noise reduction and edge detection applications. We detail the CMOS implementation of this method in the analog domain at the pixel level, leading to the development of the SIS. Additionally, we present results from post-layout simulations that evaluate the performance of the designed SIS.

3.2 Method

Kernel-based spatial filtering is equivalent to performing a two-dimensional convolution with the flipped kernel. The kernel contains the weights to be assigned to each corresponding pixel in the neighborhood of the original pixel. The convolution consists of traversing the entire image, pixel by pixel, so that each of these pixels coincides with the central pixel of the kernel, multiplying the matching values, and then summing to obtain the new value. We can calculate the filtered image using the following equation:

$$Z_i = \sum_{n \varepsilon W} w_n X_n(i), \tag{3.1}$$

where $X_n(i)$ represents the neighborhood pixels of the ith image pixel, n is a linear index running over the neighborhood region W, w_n are the corresponding kernel values, and Z_i represents the result of the kernel-based spatial filtering over the input image $X_n(i)$. The region W contains the area of the image covered by the kernel during filtering and the dimensionality of the kernel, which is $(2k + 1) \times$

$(2k + 1)$, with k being the kernel's radius, determines the size of this region and, consequently, the size of the involved variables.

Equation (3.1) may return a negative value, especially for edge detectors whose value represents the magnitude and orientation of the gradient rather than the intensity of a pixel due to kernels having both positive and negative values. Therefore, we focus only on the magnitude of the gradient, which is then represented as a pixel in an image by the absolute value calculation. The resulting value is always positive, as shown in the following equation:

$$Y_i = \left| \sum_{n \varepsilon W} w_n X_n (i) \right|. \tag{3.2}$$

where Y_i represents the outcome of kernel-based spatial filtering applied to input image $X_n(i)$, representing the magnitude of the gradient.

This work discusses three sets of filters based on Equation (3.2):

1. Filters that require a single kernel for the convolution, such as the Gaussian filter for denoising and the x-direction Sobel filter for edge detection, which generate a pixel value in their output or generate the absolute value of the gradient, respectively.

2. Filters perform two convolutions using different kernels on the same input image. The results are then compared to determine which is larger. The Sobel filter is an example.

3. Filters perform two cascaded convolutions using different kernels, with the output of the first convolution serving as the input to the second convolution. An example of such a filter is the Laplacian of the Gaussian (LoG), which is created by first applying a Gaussian filter to remove noise and then a Laplacian filter to detect edges.

We implemented Equation (3.2) to support our filter sets under the assumption that the results of spatial filters are always positive. This is because our application focuses on two main aspects: kernel-based denoising with positive coefficients and gradient magnitude calculation for edge detection. In the case of denoising, filters such as mean and Gaussian, which use positive weights, ensure that the output remains positive when applied to images. The gradient magnitude is inherently positive for edge detection regardless of the individual gradient signs.

The convolution operation is performed in parallel across all image pixels. Photodiode currents (representing the pixels) are integrated within a specified region W during a total integration time T. Although T can theoretically be any

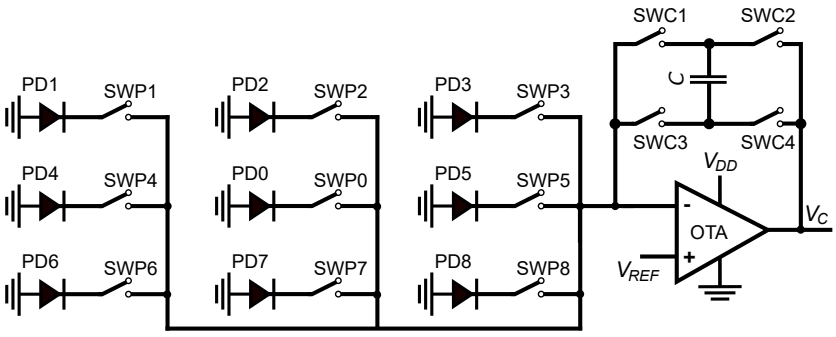


Figure 3.1: CTIA used to integrate the current of the photodiodes $PD0-PD8$, convert it into voltage, and simultaneously calculate the convolution operation.

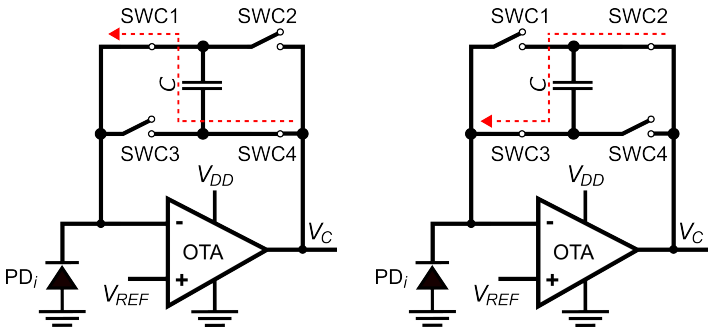


Figure 3.2: CTIA configured to integrate the photodiode currents corresponding to (**a**) positive values and (**b**) negative values of the kernel.

value, its minimum limit is defined by the response times of the circuits used in the hardware. A shorter T enables a higher frame rate, which is crucial for real-time applications. The integration time of each photodiode is proportional to the value of the associated kernel coefficient to compute the convolution according to Equation (3.2), thus avoiding addition and multiplication operations. At the end, a voltage representing the convolution result is obtained. To perform these operations, we used a CTIA [100, 101], which consists of a high-gain Operational Transconductance Amplifier (OTA), a capacitor, and four switches, as shown in Figure 3.1.

The time T is divided into two equal intervals, T_1 and T_2, a necessary condition for the proposed method's correct operation. T_1 is subdivided into intervals equal to the number of positive kernel values. The duration of each interval is directly proportional to the associated positive kernel value. The photodiode currents corresponding to these positive kernel values are integrated during each interval. Subsequently, the integrated currents are converted into voltages that increase

over time. To achieve this, switches $SWC1$ and $SWC4$ are closed, and switches $SWC2$ and $SWC3$ are opened in the CTIA shown in Figure 3.2a. This will result in a linear increase in voltage at each subinterval of T_1, starting from V_{REF}. For T_2, the procedure is similar, and we integrate the photodiode currents for negative kernel values, which are then converted to voltages. In the CTIA shown in Figure 3.2b, to achieve this, switches $SWC1$ and $SWC4$ are opened and switches $SWC2$ and $SWC3$ are closed, which causes the capacitor terminals to swap. This results in an increasing linear voltage at each subinterval of T_2 starting from V_{DD} minus the voltage reached at the end of the interval T_1. In all cases, the magnitude of the photodiode currents defines the slope values of the voltage signals.

Figure 3.3 displays two examples of the integration process of a single pixel over time T. Each example shows convolution calculations using the same kernel with positive and negative values but different photodiode current values. T_1 and T_2 are each divided into three intervals whose duration is proportional to the positive and negative kernel values, respectively. The intervals in T_1 are w_1T_1, w_3T_1, and w_5T_1, while in T_2 they are w_2T_2, w_4T_2, and w_6T_2. For instance, if the total integration time T is 16μs, and the kernel values are $w_1 = \frac{1}{8}$, $w_2 = -\frac{5}{8}$, $w_3 = \frac{1}{2}$, $w_4 = -\frac{1}{4}$, $w_5 = \frac{3}{8}$ and $w_6 = -\frac{1}{8}$, then the current for the photodiode associated with positive values w_1, w_3 and w_5 are integrated for 1μs, 4μs y 3μs, respectively. Subsequently, the current for the photodiode associated with negative values w_2, w_4 and w_6 are integrated for 5μs, 2μs and 1μs, respectively. The convolution kernel used can result in an output voltage either exceeding (red) or falling below (black) the reference voltage (V_{REF}) by the end of the integration period (T). The additional capacitor switching, illustrated by the black convolution in Figure 3.3a after T, is only required when the circuit in Figure 3.1 produces an output lower than V_{REF}, equivalent to calculating the absolute value.

A particular case occurs when the kernels consist solely of positive values, such as those found in mean and Gaussian filters used for denoising. In this scenario, the integration of the photodiode currents takes place over a duration T_1, as illustrated in Figure 3.3b, resulting in a higher frame rate that of Figure 3.3a. The time T_1 is divided into five intervals, each with a duration proportional to the corresponding kernel values: w_0T_1, w_7T_1, w_8T_1, w_9T_1, and $w_{10}T_1$. For instance, if the total integration time T is 16μs, and the kernel values are $w_0 = \frac{1}{2}$ and $w_7 = w_8 = w_9 = w_{10} = \frac{1}{8}$, then the current for the photodiode associated with w_0 is integrated for 4μs. The other four photodiodes corresponding to w_7, w_8, w_9, and w_{10} integrate for 1μs each. Since all kernel values are positive, the resulting voltage will consistently increase, ensuring it remains greater than V_{REF}.

For better understanding, we divide the kernel values in the region W of Equation (3.2) into two subregions, W_1 and W_2, with all values positive and negative, respectively, as shown in Equation (3.3).

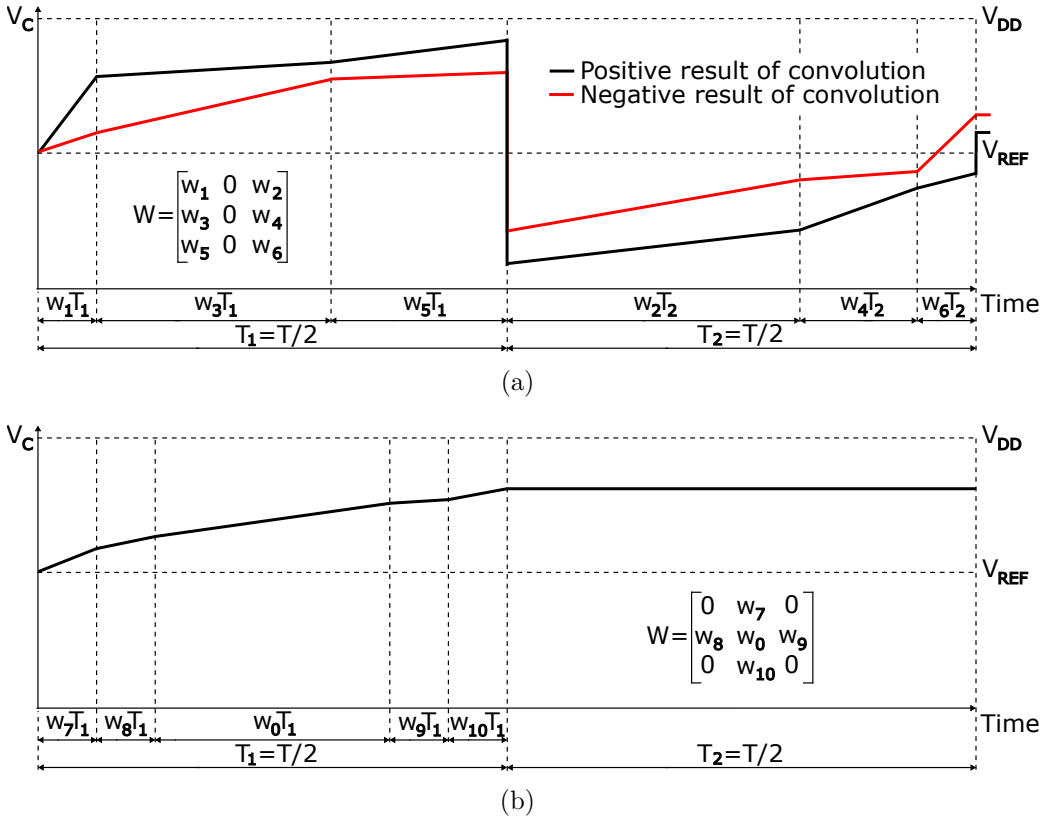


Figure 3.3: Convolution method proposed with (**a**) positive (w_1, w_3, and w_5) and negative (w_2, w_4, and w_6) kernel values and (**b**) positive kernel values.

$$Y_i = \left| \sum_{n \varepsilon W_1} w_n X_n\left(i\right) + \sum_{n \varepsilon W_2} w_n X_n\left(i\right) \right| \tag{3.3}$$

Several edge detection filters, such as Robert, Prewitt, and Sobel, utilize normalized kernels with both positive and negative values. The sum of the positive values equals 1, and the sum of the absolute values of the negative values also equals 1. Each pixel in the image corresponds to a function of the respective photodiode current, expressed as $X_n(i) = \frac{X_{max}}{I_{max}} I_n(i)$, as shown in the following equation:

$$Y_i = \frac{X_{max}}{I_{max}} \left| \sum_{n \varepsilon W_1} w_n I_n\left(i\right) + \sum_{n \varepsilon W_2} w_n I_n\left(i\right) \right|, \tag{3.4}$$

when I_{max} represents the maximum current of the photodiode, X_{max} represents the maximum intensity value of the image X, and $I_n(i)$ represents the photodiode current.

The circuit depicted in Figure 3.2, when it closes two opposing switches, and the others remain open, thereby creating a negative feedback loop, becomes an inverting integrating amplifier with a reference voltage. The term $\sum_{n\varepsilon W_1} w_n I_n(i)$ of Equation (3.4) is present in the transfer function of this circuit, as shown in the following equation:

$$V_C^{'}(i) = \frac{T_1}{C} \sum_{n\varepsilon W_1} w_n I_n(i) + V_{REF}, \tag{3.5}$$

where T_1 is the integration time of the photodiode currents that match with the positive values of the kernel W, C is the integration capacitor, $V_C^{'}$ is the output voltage at the end of the calculation using values within W_1, and $V_{REF} = \frac{V_{DD}}{2}$ guarantees that the OTA output signal is symmetrical and has a maximum dynamic range. V_{DD} is the supply voltage.

A second integration process, initiated from the voltage $V_C^{'}$, yielded Equation (3.6), which included the term $\sum_{n\varepsilon W_2} w_n I_n(i)$ of Equation (3.4).

$$V_C(i) = -\frac{T_2}{C} \sum_{n\varepsilon W_2} w_n I_n(i) + V_{DD} - V_C^{'}(i) \tag{3.6}$$

T_2 is the integration time of the photodiode currents that match with the negative values of the kernel W and V_C is the output voltage at the end of T_2.

Subsequently, by substituting Equation (3.5) into Equation (3.6) and considering $T_1 = T_2 = \frac{T}{2}$, we can deduce Equation (3.7), which enables the computation of the convolution operation on an image using the circuit depicted in Figure 3.1.

$$V_C(i) = V_{REF} - \frac{T}{2C} \left[\sum_{n\varepsilon W_2} w_n I_n(i) + \sum_{n\varepsilon W_1} w_n I_n(i) \right] \tag{3.7}$$

By rearranging Equation (3.7), we derive Equation (3.8), which illustrates that the convolution operation can be executed by controlling the integration time of each photodiode.

$$V_C(i) = V_{REF} - \frac{1}{C} \left[\sum_{n\varepsilon W_2} (w_n T_2) I_n(i) + \sum_{n\varepsilon W_1} (w_n T_1) I_n(i) \right] \tag{3.8}$$

if we develop algebraically the Equation (3.7), we get:

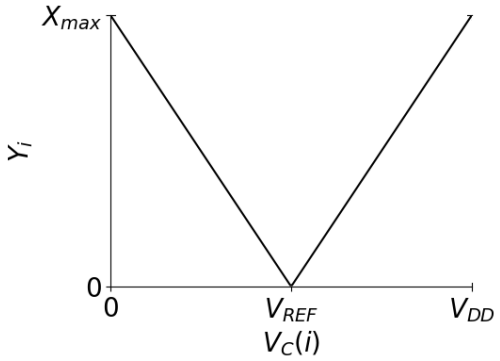


Figure 3.4: Relationship between the CTIA output voltage and the resulting pixel value of the kernel-based image filter.

$$\left[\sum_{n\varepsilon W_2} w_n I_n(i) + \sum_{n\varepsilon W_1} w_n I_n(i)\right] = \frac{2C}{T}\left[V_{REF} - V_C(i)\right] \tag{3.9}$$

$$\left|\sum_{n\varepsilon W_2} w_n I_n(i) + \sum_{n\varepsilon W_1} w_n I_n(i)\right| = \frac{2C}{T}\left|V_{REF} - V_C(i)\right| \tag{3.10}$$

$$\frac{X_{max}}{I_{max}}\left|\sum_{n\varepsilon W_2} w_n I_n(i) + \sum_{n\varepsilon W_1} w_n I_n(i)\right| = \frac{2CX_{max}}{TI_{max}}\left|V_{REF} - V_C(i)\right| \tag{3.11}$$

Comparing Equation (3.4) and Equation (3.11), we get:

$$Y_i = \frac{2CX_{max}}{TI_{max}}|V_{REF} - V_C(i)| \tag{3.12}$$

In order to guarantee that, if $V_C(i)$ is equal to either 0 or V_{DD}, the output pixel value will be X_{max}, and that, in the case where $V_C(i)$ is equal to V_{REF}, the output value pixel will be 0; we define $V_{REF} = \frac{TI_{max}}{2C}$. Consequently, we obtain the Equation (3.13) and its graphic representation in Figure 3.4.

$$Y_i = \frac{X_{max}}{V_{REF}}|V_{REF} - V_C(i)| \tag{3.13}$$

Figure 3.4 shows the relationship between the CTIA output voltage shown in Figure 3.1 and the resulting value of the kernel-based image filter represented as a pixel. The CTIA output voltage can take values between 0 and V_{DD}, while the value resulting from the filter can take values between 0 and X_{max}.

3.3 SIS architecture

The designed Smart Image Sensor integrates a kernel-based spatial filter for denoising and edge detection at the pixel level in the analog domain. This implementation contributes to decreasing power consumption and increasing frame rates, although it increases the fill factor of the imager (i.e., the ratio between the die area occupied by the photodetector and the area of the entire pixel circuit).

3.3.1 Kernel-based spatial filter circuit

The proposed kernel-based spatial circuit is part of our smart pixel implemented at the pixel level in the analog domain. Each smart pixel in the SIS comprises a photodiode and a circuit that performs spatial filtering using a 3×3 kernel (Figure 3.5). This circuit consists of a custom CTIA (See Section 3.2), Sample & Hold circuit, CTIA switches control circuit, maximum selector circuit, and voltage to current converter. The CTIA receives current from photodiode PD_0 and its neighboring photodiodes PD_1 to PD_8. Additionally, it receives a current value from each of its eight neighbors corresponding to their values obtained in a previous spatial filtering. The latter currents are only necessary in the third set of filters presented in Section 3.2. The SIS generates an output voltage V_0, which corresponds to the absolute value of the spatial filtering result.

The CTIA consists of a two-stage OTA with high gain [100, 101], a capacitor and four switches controlled by the outputs of the CTIA switches control circuit (signal $SW1'$ for the switches $SWC1$ and $SWC4$, and signal $SW2'$ for the switches $SWC2$ and $SWC3$). We use the CTIA circuit because of its high precision, wide dynamic range, high linearity, and high SNR [102, 103]. This configurable architecture makes it possible, by switching the switches, to integrate the photodiode current in one of the two possible directions for a required time and to produce a voltage that represents the convolution made on the intensity of the light captured by its photodiode and the photodiodes of its neighbors.

The circuit shown in Figure 3.5 has three modes of operation corresponding to the three sets of filters presented in Section 3.2. In all cases, the CTIA performs the integration process in two phases of equal duration, the total time being T. The first phase integrates the currents generated by the photodiodes associated with the positive values of the kernel ($SWC1$ and $SWC4$ open, and $SWC2$ and $SWC3$ closed). The second phase integrates the currents generated by the photodiodes associated with the negative values of the kernel ($SWC1$ and $SWC4$ closed, and $SWC2$ and $SWC3$ open).

For better understanding, this method will denote each pixel within the 3×3 distribution $P0 - P8$, as shown in Figure 3.1.

Figure 3.6 shows the activation of the switches $SWP3$, $SWP5$, and $SWP8$

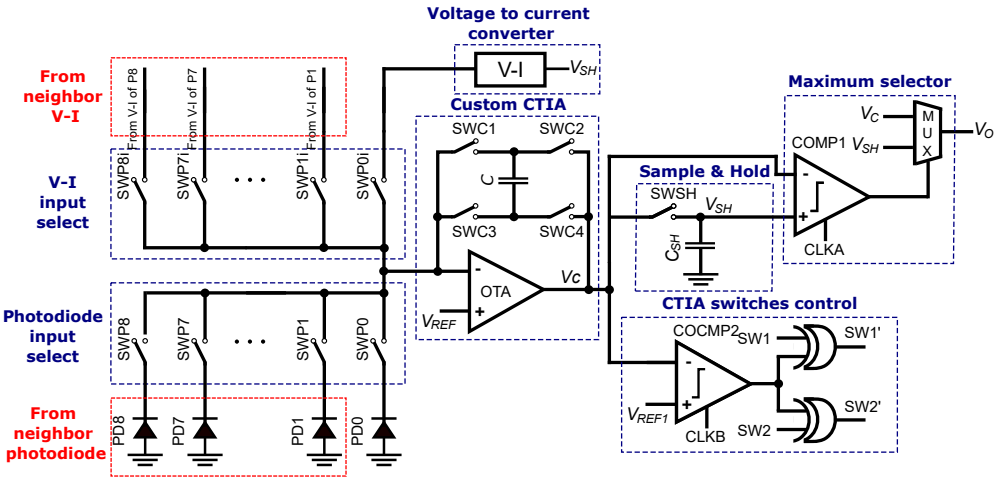


Figure 3.5: Spatial filtering circuit capable of performing single kernel-based, different kernel-based operations on the same input image, and cascaded kernel-based operations.

during $T/8$, $T/4$, and $T/8$, respectively, in the first phase, according to the kernel in the x-direction of the Sobel filter (See Figure 3.15g). In the second phase, switches $SWP1$, $SWP4$, and $SWP6$ are activated during $T/8$, $T/4$, and $T/8$, respectively. The duration of both phases is $T/2$. First, V_{REF1} is set to 0 V, which sets the output of comparator $COMP2$ to logic 0. External signals $SW1$ and $SW2$ then control the CTIA switches. The value of V_{REF1} changes to $V_{DD}/2$ before the end of the second phase. At the end of the second phase, the output of comparator $COMP2$ is set to logic 1 if $V_C > V_{REF1}$. This results in a logical inversion of the signals $SW1'$ and $SW2'$, which exchange the terminals of the integration capacitor C. The operation ensures the calculation of the absolute value of the spatial filter output. The capacitor C_{SH} is discharged by keeping the switch $SWSH$ open. Thus, the comparator $COMP1$ outputs logic 0, enabling the CTIA output (V_C) to pass through the analog multiplexer to the circuit output V_0. The remaining input and all V-I input switches remain open. If the kernel values are positive, the total integration time is halved because there is no second phase. In addition, V_C is always greater than V_{REF}, so swapping the integration capacitor C terminals is unnecessary.

The Sobel filter computation relies on the activation of switches $SWP1$-$SWP8$, illustrated in Figure 3.7. This Figure also depicts the outputs of the CTIA (V_C) and spatial filtering circuit (V_0). The Sobel kernels are used independently in the x- and y-directions but with the same pixel window of the image. The circuit first calculates the x-direction Sobel filter as a single kernel (See Figure 3.15g). At the end of its second phase, as V_C is lower than V_{REF1}, comparator $COMP2$ outputs a logic 1, and the signals $SW1'$ and $SW2'$ are inverted, resulting in a change in the

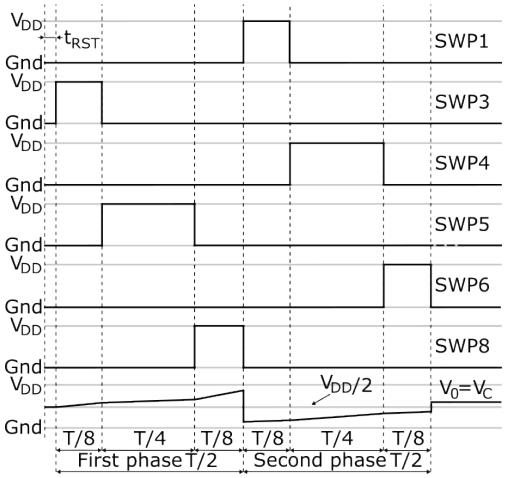


Figure 3.6: Input select switches states and output voltage of the circuit for computing the x-direction Sobel filter.

polarity of the integration capacitor C. Thus, it guarantees that V_C is greater than V_{REF1} due to the comparator output changing to a logic 0, while the signals $SW1'$ and $SW2'$ remain unaltered. The Sample & Hold circuit stores the value V_C in the capacitor C_{SH} for a time T_{SH}. Once the value is stored, comparator $COMP2$ and CTIA are reset, and capacitor C is discharged. The circuit then calculates the Sobel filter in the y-direction as a single kernel (See Figure 3.15h) using the same method. Comparator $COMP2$ performs the same comparison as before to ensure that V_C is greater than V_{REF1}. Finally, the maximum selector circuit determines which of the x-direction or y-direction filter results is the largest. If $V_{SH} > V_C$, the x-direction filter result is greater; otherwise, the filter result in the y-direction is greater. As in the previous case, the rest of the input and none of the V-I input selector switches are unused.

The LoG filter applies the modified Laplacian filter to an image filtered with a Gaussian filter, using the kernels shown in Figure 3.15i and Figure 3.15b, respectively. The circuit first calculates the Gaussian filter for a duration of $T/2$, which is determined by the external signals $SW1$ and $SW2$, which have values of 1 and 0, respectively. Figure 3.8 displays the activation of the switches $SWP0$-$SWP8$ necessary for the computation of the Gaussian filter. The voltages resulting from this filter are always greater than $V_{REF1} = 0$. During this phase, none of the V-I input selectors are used. The voltage is stored in capacitor C_{SH} using a Sample & Hold circuit by closing switch $SWSH$ for a duration of T_{SH}. Later, the CTIA circuit is reset, and the V-I circuit converts the stored voltage into current. As a result, each spatial window filter circuit has its current, and when combined with the currents of neighboring circuits, it is possible to calculate the modified Laplacian filter. Figure 3.8 also illustrates the activation of switches $SWP0i$ dur-

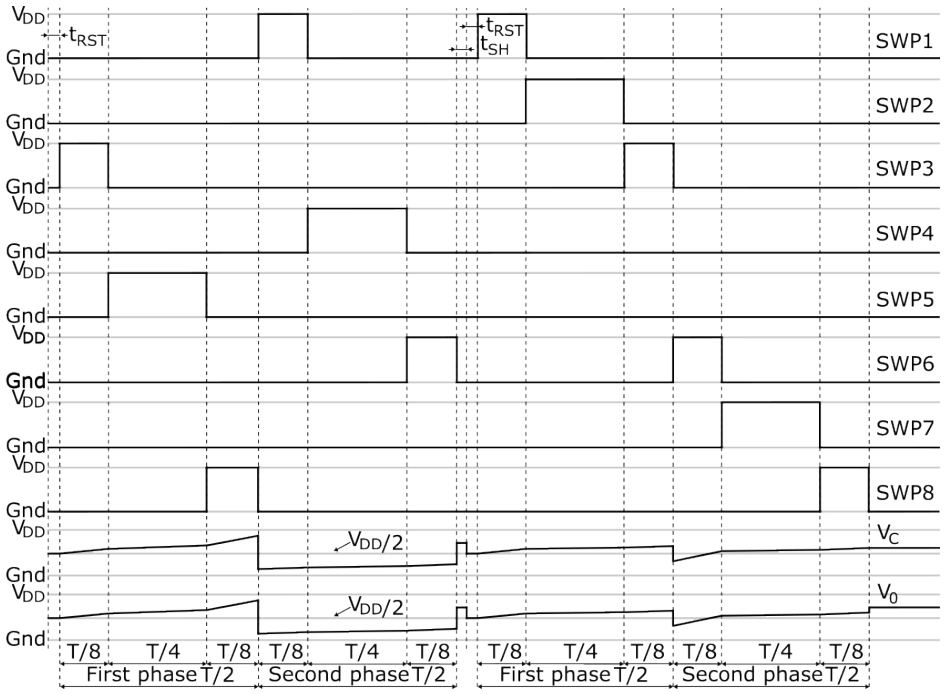


Figure 3.7: Input-select switches states, output voltage of the CTIA and output voltage of the circuit for computing the full Sobel filter.

ing $T/2$ in the first phase, according to the kernel of the modified Laplacian filter (See Figure 3.15i). In the next phase, the switches $SWP2i$, $SWP4i$, $SWP5i$, and $SWP7i$ are sequentially activated at $T/8$. The CTIA switches are directly controlled by signals $SW1$ and $SW2$ due to $V_{REF1} = 0$ and the output of comparator $COMP2$ being a logic 0. The capacitor C_{SH} is intentionally discharged to display the value V_C at the output of the Maximum selector circuit. The input selector switches remain open during this final filtering process.

The schematic diagram of the two comparators used in the proposed kernel-based spatial filtering circuit is shown in Figure 3.9. Both are discrete-time (dynamic) CMOS comparators with two stages: latch-type sense amplifier and RS-NAND latch. They have high input impedance, high speed, and low power. When $CLK = 0$, V_{OP} and V_{ON} inputs of the RS-NAND latch are driven to V_{DD}, with M_1 off and $M_8 - M_9$ on. During this state, the output of the RS-NAND circuit remains unchanged. When $CLK = V_{DD}$, M_1 is on while $M_8 - M_9$ are off. V_{OP} and V_{ON} start to decrease from V_{DD} at different rates depending on the corresponding input voltages (V_{IP} and V_{IN}). Assuming that $V_{IP} > V_{IN}$, the voltage V_{OP} decreases faster than V_{ON}, reaching the value of $V_{DD} - V_{thp}$. When M_6 turns on, it initializes latch generation through two cross-coupled inverters. As a result, V_{ON} becomes V_{DD} and V_{OP} falls to ground. Thus, it causes the R-S NAND, and

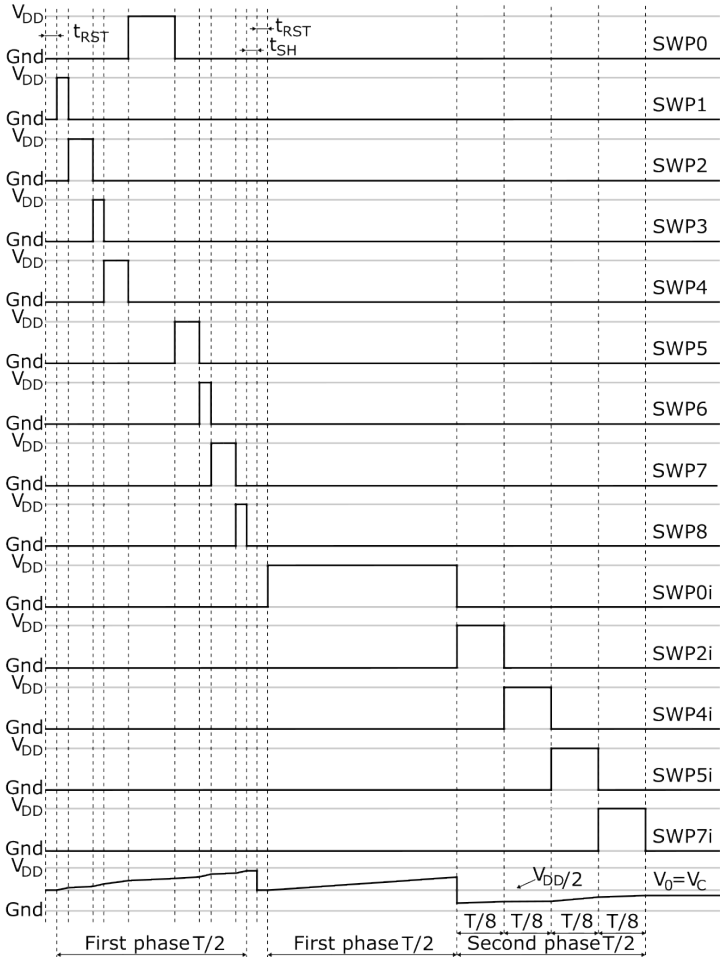


Figure 3.8: Input select and V-I input select switches states and output voltage for computing the LoG filter.

consequently the full circuit, to show a logic 1 at the output V_0. A similar analysis applies when $V_{IP} < V_{IN}$, resulting in a logic 0 at the output of the circuit.

The circuit depicted in Figure 3.10 is a modified version presented in [104], designed to achieve the required voltage and current ranges. The PMOS transistor M_3 performs the voltage-to-current conversion. M_1 and M_2 form a damping circuit to increase the linear input voltage range. The remaining circuit consists of current mirrors, which enable the desired output current range to be obtained by modifying their geometry.

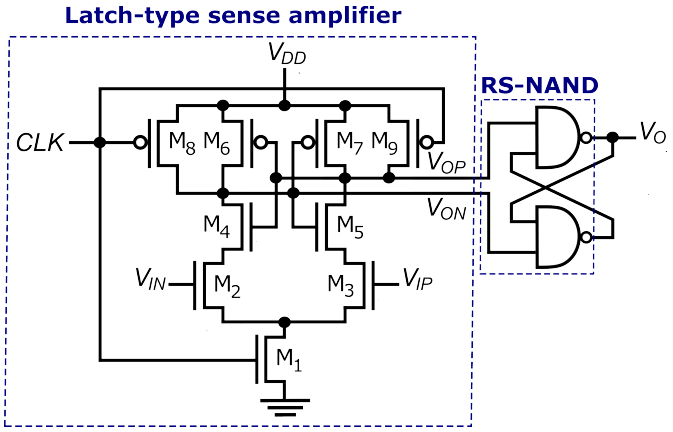


Figure 3.9: Discrete-time CMOS comparator.

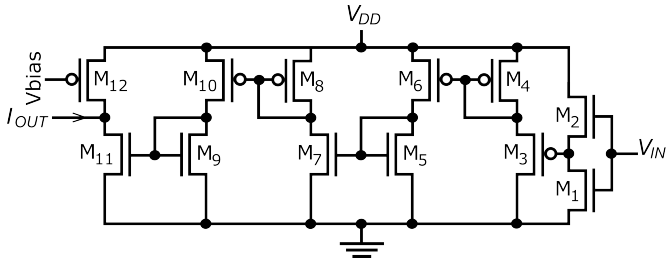


Figure 3.10: Voltage-to-current converter circuit.

3.3.2 General readout circuit

Figure 3.11 displays the SIS, which consists of a 128 × 128 smart pixel array and other modules such as Row select & control signal, Analog column multiplexer, External digital device, and Output stage. Each smart pixel comprises a photodiode and a kernel-based spatial filter circuit. The smart pixel array and the analog column multiplexer that generate the output require precise timing and control signals provided by a dedicated External digital device.

Each smart pixel receives 22 signals that directly control the switches in the circuit. These signals include photodiode input selection (9 signals), V-I input (9 signals), CTIA (2 signals), Sample & Hold (1 signal), and row selection (1 signal). Additionally, each smart pixel has an output corresponding to the filtered voltage generated by its kernel-based spatial filter circuitry. After completing the voltage calculations within each pixel, the 128 rows are sequentially activated. The analog column multiplexer enables serial access and output of individual pixel voltage values, resulting in 128 values.

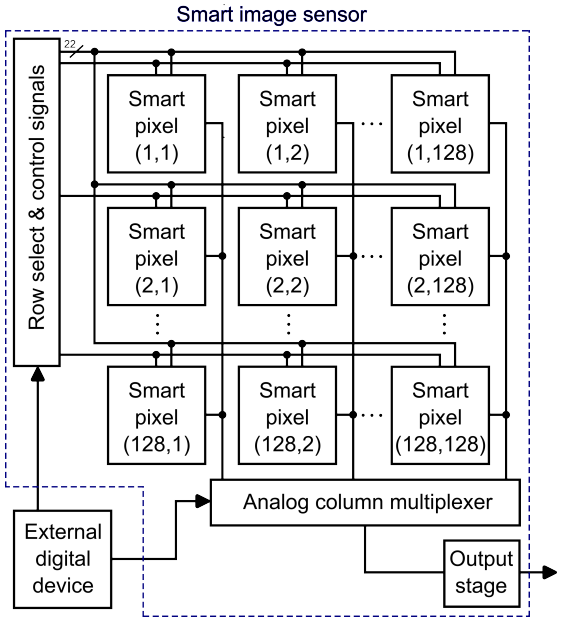


Figure 3.11: SIS controlled by External digital device.

3.4 Results

3.4.1 Physical layout

The Electronic Design Automation (EDA) tools used in this work include LTSpice for simulations and the Electric VLSI Design System for layout and verification. These tools support various verification processes, such as Design Rule Checking (DRC), Electrical Rule Checking (ERC), and Network Consistency Checking (NCC). We performed a physical design of the kernel-based spatial filtering circuit in a 0.35 μm mixed-signal CMOS process with two poly layers, three metal layers, and a 3.3 V supply voltage, as shown in Figure 3.12.

With a single-phase integration time of 9 μs, a maximum photodiode current of 18 nA, and a double-poly capacitor with a capacitance per unit area of 950 aF/μm^2, the pixel necessitates a 100 fF integration capacitor measuring 13.1 μm by 8.0 μm.

The fill factor is a crucial metric when evaluating the impact of adding computing capacity to a pixel. A high fill factor in an image sensor improves higher light sensitivity, a better SNR, and higher image resolution, resulting in sharper and more detailed images in a different lighting conditions [30, 21, 105]. The spatial filter circuit has an area of 63.4 μm by 51.3 μm. With a pixel size of 64 μm by 64 μm [106], the circuit achieves a fill factor of 20.6 %. This means the area occupied by the circuit within the pixel is approximately four times larger

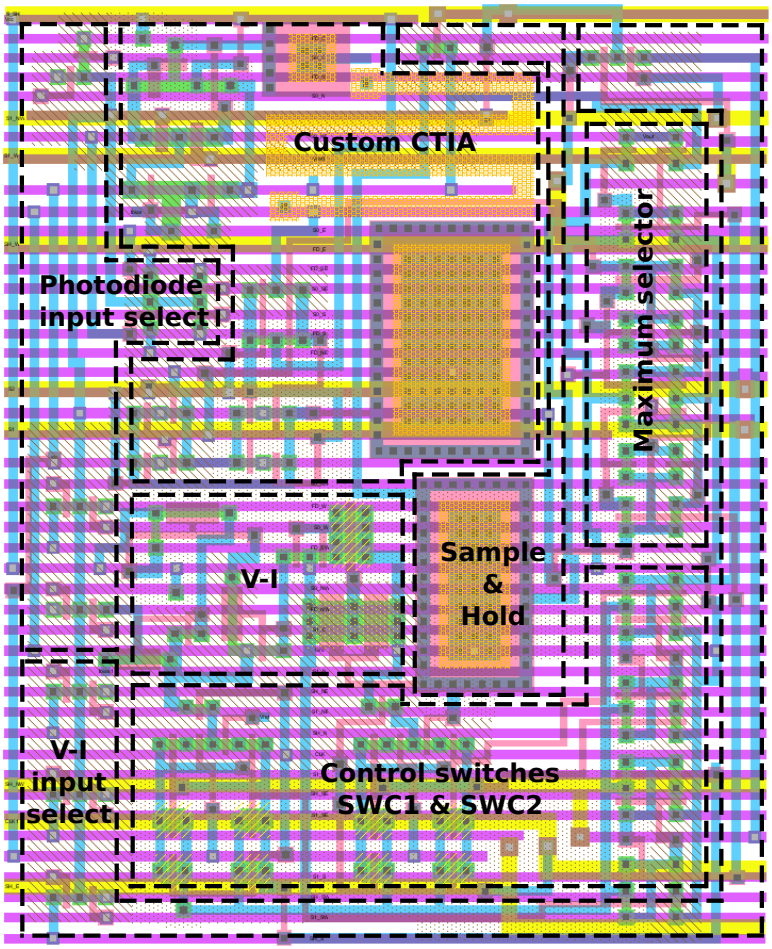


Figure 3.12: Layout of kernel-based spatial filter circuit.

than that of the photodetector. While this fill factor is relatively low, it is typical for CIS designed with 0.35 μm technology that incorporate pixel-level computation. For instance, reported fill factor values in the literature include 16.3 % [107], 5.7 % [108], 12.6 % [109], 17.7 % [110], and 2.7 % [106]. The pixel containing only the CTIA, which does not allow any computation but only performs the integration of the photocurrent, has a fill factor of 70.4 %. If the goal is to enhance this pixel by adding computational capacity for convolution with a single kernel, only eight input switches are needed to connect with neighboring pixels. However, this modification reduces the fill factor to 68.6 %.

Although the results presented in this section are based on the physical design in a 0.35 μm process, in order to assess scalability, the design was adapted to a standard 0.18 μm process with a supply voltage of 1.8 V. This design included Metal Insulator Metal (MIM) capacitors [24, 79, 73, 80, 19], which exhibit a

capacitance per unit area of 2 fF/μm^2. The spatial filter circuit in the 0.18 μm process has a total area of 32.9 μm by 26.8 μm, achieving a fill factor of 78.4 % in the same 64 μm by 64 μm size pixel. When considering only the CTIA, the fill factor is 92.6 %, while when adding the eight input switches that connect with the neighbors, the fill factor decreases to 91.5 %.

Post-layout kernel-based spatial filter circuit simulations require an integration time of 9 μs plus 50 ns to reset the integration capacitor after each single convolution. The time required by the comparators and Sample & Hold circuits is 0.1 μs. The complete circuit operates in parallel for all pixels in the image. During readout, the row-select circuit exhibits a delay of 60 ns, and the column multiplexer circuit has a delay of 80 ns. Taking the Sobel of the Gaussian (SoG) filter as a reference, which is the filter that requires the most computational time due to its three-kernel operation, the readout time for the complete array is 1.34 ms, allowing for the processing of images at a maximum frame rate of 743 fps.

3.4.2 Post-layout simulation

Two hundred images with varying degrees of brightness, contrast, and DR were randomly selected from the Linnaeus 5 dataset [1] for evaluating our kernel-based spatial filter circuit. All images have a resolution of 128×128 pixels, which aligns with the dimensions of the proposed CMOS readout circuit array. In the case of the mean, Gaussian, LoG, and SoG filters, Gaussian noise with a standard deviation of 0.01 was added to the images.

Figure 3.13 illustrates a post-layout simulation of the proposed spatial filter configured as Sobel filter. We selected the Sobel filter because, as explained in Section 3.2, it is present in the three sets of filters. The graph shows three output voltage signals: the CTIA (V_C) in blue, the Sample & Hold circuit (V_{SH}) in green, and the spatial filter (V_0) in red. Initially, the circuit enters a reset state for 50 ns, resulting in the outputs of the CTIA and SIS being $V_{DD}/2 = 1.65V$. During the next 18 μs, Sobel filtering is conducted in the x-direction. At the end of this stage, the resulting voltage is stored in the Sample & Hold capacitor, followed by another reset. Subsequently, the circuit performs Sobel filtering in the y-direction for another 18 μs, and it compares the result to the previously stored value. Finally, the spatial filter outputs the higher of the two voltages.

Moreover, the time required for the comparators and Sample & Hold circuits is 0.1 μs. The complete circuit operates efficiently in parallel for all pixels in the image. During readout, the row-select circuit introduces a delay of 60 ns, while the column multiplexer circuit adds a delay of 80 ns. Referencing the denoising filter (mean or Gaussian) and SoG filter, which require the lowest and highest computational times, respectively, the total readout times for the complete array are 1.33 ms and 1.34 ms. This allows image processing at frame rates ranging

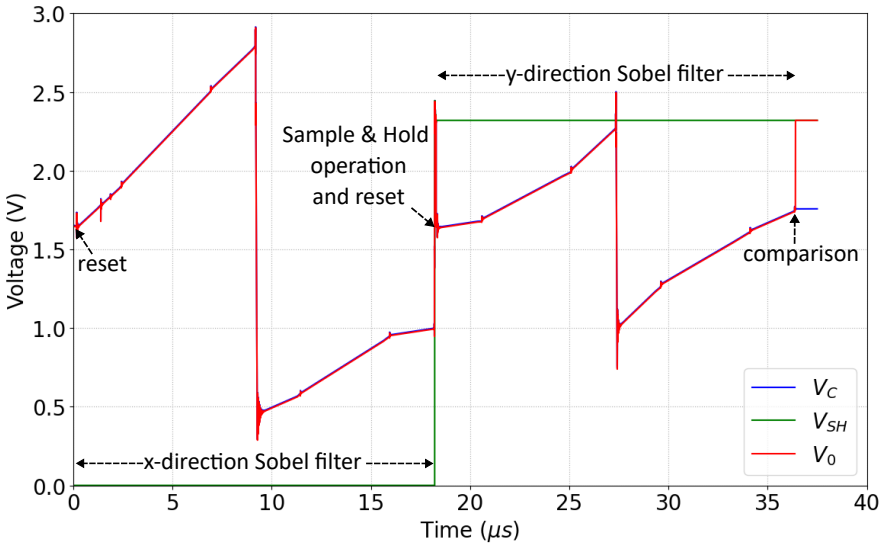


Figure 3.13: Post-layout simulation of the proposed kernel-based spatial filter configured as a Sobel filter for edge detection. The voltage signals displayed are the output of the CTIA (V_C) in blue, the output of the Sample & Hold circuit (V_{SH}) in green, and the output of spatial filter (V_0) in red.

from 743 to 752 fps.

On the other hand, when you configure the proposed SIS as a LoG or SoG filter, the V-I circuit shown in Figure 3.10 significantly affects its performance. Figure 3.14 shows the simulated and ideal DC transfer characteristics of the V-I circuit using blue and red colors, respectively. According to the DC response, we use two different errors to validate this circuit: the output error is the deviation of the output current at each point from its ideal value, and the linearity error is the deviation of a straight line passing through the simulated V-I characteristic points.

The output error of our voltage-to-current converter circuit is 5.1 %, and the linearity error is 2.0 %. While some reported designs have achieved output errors below 2.5 % and linearity errors under 0.8 % [111, 112, 113], our circuit utilizes fewer transistors and generates a narrower output current range in the nanoampere (nA) range. This tradeoff between error rates and circuit complexity emphasizes the advantages of our design.

3.4.3 Circuit performance

In this section, we will evaluate the performance of our CIS operating as various kernel-based spatial filters. We show the result of applying our spatial filter to images taken from the Linnaeus 5 dataset [1], some of which we added Gaussian

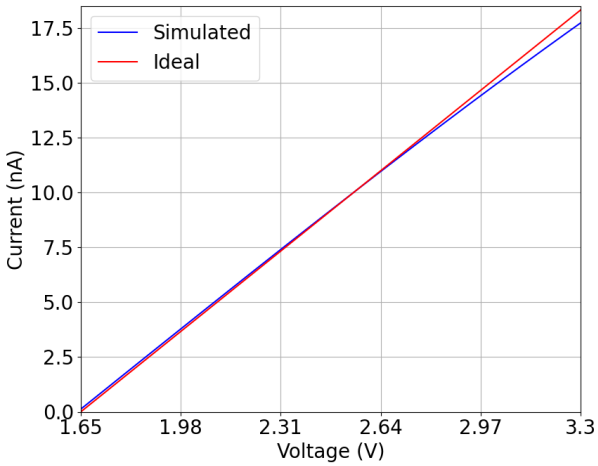


Figure 3.14: Simulated and ideal DC transfer characteristic of the V-I circuit.

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \quad \frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} \quad \begin{bmatrix}0&0&-1\\0&1&0\\0&0&0\end{bmatrix} \quad \begin{bmatrix}-1&0&0\\0&1&0\\0&0&0\end{bmatrix} \quad \frac{1}{3}\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$$

(a) (b) (c) (d) (e)

$$\frac{1}{3}\begin{bmatrix}1&1&1\\0&0&0\\-1&-1&-1\end{bmatrix} \quad \frac{1}{4}\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix} \quad \frac{1}{4}\begin{bmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{bmatrix} \quad \frac{1}{4}\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix} \quad \frac{1}{8}\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}$$

(f) (g) (h) (i) (j)

Figure 3.15: 3×3 filter kernels used for conducted tests: (**a**) mean, (**b**) Gaussian, (**c**) x-direction Robert, (**d**) y-direction Robert, (**e**) x-direction Prewitt, (**f**) y-direction Prewitt, (**g**) x-direction Sobel, (**h**) y-direction Sobel, (**i**) modified Laplacian, and (**j**) Laplacian.

noise with a standard deviation of 0.01. To evaluate the performance of the hardware implementation, we will compare it with its software implementation using the same kernel. We use the eight 3×3 kernels shown in Figure 3.15. In order to apply our spatial filter to an input image and generate an output image, we employed the following methodology. We converted the pixel values in the input image to currents ranging from 0 to 18 nA, which simulated the operation of a photodiode. Subsequently, we conducted a post-layout Spice simulation of the circuits to produce the pixel values required to form the output image using the circuit. Finally, the output pixel voltages ranged from 0 to 3.3 V and were converted to values between 0 and 255 to display the output images.

First, we will evaluate the mean and Gaussian filters whose kernels are shown in Figures 3.15a and 3.15b. Figure 3.16 illustrates the images utilized in the

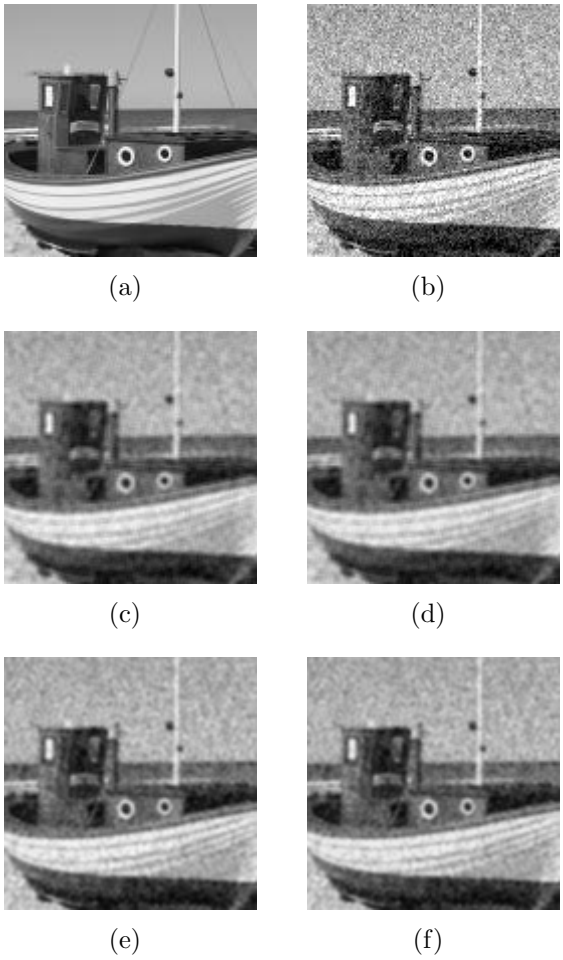


Figure 3.16: Noise reduction using a 3×3 kernel-based spatial filter circuit: (**a**) original image (from Linnaeus 5 dataset [1]), (**b**) image with added Gaussian noise, (**c**) software-based mean filter, (**d**) hardware-based mean filter, (**e**) software-based Gaussian filter, and (**f**) hardware-based Gaussian filter.

3×3 kernel-based spatial filter circuit operation for noise removal. Figure 3.16a is the reference image taken from the Linnaeus 5 dataset [1], while Figure 3.16b displays the result of contaminating the reference image using Gaussian noise with a standard deviation of 0.01. Figures 3.16c and 3.16e display the outputs of the mean and Gaussian spatial filters, respectively, implemented in software using the kernel presented in Figure 3.15a and 3.15b. Meanwhile, Figures 3.16d and 3.16f show the output images of the mean and Gaussian spatial filters implemented by the proposed circuit using the same kernel. A visual inspection of the resulting images of the proposed circuit (Figures 3.16d and 3.16f) indicates that they are visually indistinguishable from their corresponding software versions.

To evaluate the results quantitatively, we employed three image quality metrics: Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) [114]. In all cases, the subscripts R and 0 refer to the images resulting from the software and hardware spatial filter computations, respectively. The software-generated image is the reference image, while the hardware-generated image is the tested image. The MSE is calculated by averaging the squared pixel intensity differences between the reference and test images and can be calculated by the following equation:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(X_0(i,j) - X_R(i,j) \right)^2, \tag{3.14}$$

where X_R is the reference image, X_0 is the tested image, and M and N are the dimensions of the images.

The PSNR metric is defined based on the pixel with the highest intensity value in the image and the MSE, as shown in the following equation:

$$PSNR = 10 \times log \frac{(2^n - 1)^2}{MSE}, \tag{3.15}$$

where n is the number of bits used to encode the intensity value of a pixel.

The SSIM is a metric used to assess the quality of images by measuring their similarity based on luminance, contrast, and structure, as shown in the following equation:

$$\begin{aligned} SSIM = & \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{2\mu_0(i,j)\,\mu_R(i,j) + C_1}{\mu_0^2(i,j) + \mu_R^2(i,j) + C_2} \\ & \times \frac{2\sigma_0(i,j)\,\sigma_R(i,j) + C_1}{\sigma_0^2(i,j) + \sigma_R^2(i,j) + C_2} \\ & \times \frac{\sigma_{0,R}(i,j) + C_3}{\sigma_0(i,j)\,\sigma_R(i,j) + C_3}, \end{aligned} \tag{3.16}$$

where C_1, C_2 and C_3 are small constants, μ_0 and μ_R are the sample spatial standard deviation at the ijth element of the resulting hardware and reference image, σ_0 and σ_R are the sample spatial standard deviation at the ijth element of the tested and reference image, and the $\sigma_{0,R}$ is the sample cross-covariance at the ijth element of the covariance image between resulting hardware and reference image.

Table 3.1 compares MSE, PSNR, and SSIM performance metrics for spatial filters that remove Gaussian noise. Compared to other denoising filters in the literature, our proposed circuit performance was validated using post-layout Spice

Table 3.1: Comparison of MSE, PSNR. and SSIM performace metrics of spatial filters for removing the Gaussian noise.

Filter	Resolution	MSE	PSNR	SSIM
[34] (mean)	-	4.74	41.37	0.99
[34] (binomial)	-	3.02	43.32	0.99
[35]	275×300	11.64	37.47	-
[115]	100×100	4.87	41.25	0.85
[116]	28×28	241.59	24.30	0.80
mean*	128×128	0.90	48.55	0.99
Gaussian*	128×128	0.98	48.19	0.99

*Proposed.

simulations for both mean and Gaussian filter configurations. The experiment utilized 200 randomly selected 128×128-pixel images from the Linnaeus 5 dataset [1], with Gaussian noise added at a standard deviation 0.01. The reference and test images were obtained using different software and hardware. Our mean and Gaussian filters achieved a significantly lower MSE (3.35x and 3.08x smaller, respectively) and higher PSNR (5.23 dB and 4.87 dB larger, respectively) compared to other works. Based on the SSIM metric, our results match those of other works with a value of 0.99. This is a positive outcome because the integration capacitor of the CTIA is charged linearly using photodiode currents, and with the voltage V_{REF} at the non-inverting input of the CTIA, it is possible to estimate and compensate for the impact of the charge injection received by the integration capacitor.

Additionally, we evaluate our proposed spatial filter for edge detection using a 3×3 single kernel, as presented in Figure 3.18. Figure 3.17 is the reference image from the Linnaeus 5 dataset [1]. The output images of the x-direction Roberts, y-direction Roberts, x-direction Prewitt, y-direction Prewitt, x-direction Sobel, y-direction Sobel, modified Laplacian and Laplacian spatial filters implemented in software using the kernel presented in Figure 3.15c- 3.15j are shown in Figure 3.18a- 3.18h, respectively. Figure 3.18i- 3.18p show the output images of the x-direction Roberts, y-direction Roberts, x-direction Prewitt, y-direction Prewitt, x-direction Sobel, y-direction Sobel, Laplacian and modified Laplacian spatial filters implemented by the proposed circuit using their respective kernels.

The images produced by the proposed circuit, configured as x-direction and y-direction Roberts filters and shown in Figure 3.18i and Figure 3.18j, respectively, exhibit a high similarity to their software-generated counterparts. In the remaining images obtained from the proposed circuit, as illustrated in Figure 3.18, there is a slight increase in pixel intensity, noticeably visible in areas that should

Figure 3.17: Original image (from Linnaeus 5 dataset [1] used by spatial filter circuit for edge detection with 3×3 single kernel.

appear black. These regions are highlighted using rectangles with colored edges: dark blue for the hardware and software images resulting from the x-direction Prewitt filter, light blue for those from the y-direction Prewitt filter, dark green for the x-direction Sobel filter, light green for the y-direction Sobel filter, dark magenta for the Laplacian filter, and light magenta for the modified Laplacian filter. This visual discrepancy primarily arises from charge injection into the integration capacitor when the input switches are activated, occurring multiple times during both integration phases. Nevertheless, an acceptable similarity remains between the hardware and software images.

Figure 3.19 illustrates the images involved in operating the spatial filter circuit for edge detection using two 3×3 kernels in the x- and y-directions: Roberts, Prewitt, and Sobel. The Roberts filter necessitates the kernels depicted in Figure 3.15c and Figure 3.15d, the Prewitt filter relies on the kernels from Figure 3.15e and Figure 3.15f, and the Sobel filter utilizes the kernels from Figure 3.15g and Figure 3.15h. The original image is shown in Figure 3.18a. The output images of the Roberts, Prewitt, and Sobel spatial filters are shown in Figures 3.19a, 3.19b, and 3.19c, respectively. Figures 3.19d, 3.19e, and 3.19f show the output images of the Roberts, Prewitt, and Sobel spatial filters, respectively, obtained by the proposed circuit.

The images used in the operation of the spatial filter circuit for edge detection, as depicted in Figure 3.20, include those generated by cascaded 3×3 kernels, comprised of Laplacian of the Gaussian and Sobel of the Gaussian filters. The first kernel employed by both is the Gaussian kernel, as illustrated in Figure 3.15b. The second kernel utilized by the LoG filter is depicted in Figure 3.15i. In contrast, the SoG filter necessitates the inclusion of two additional kernels, as shown in Figure 3.15g and Figure 3.15h. The original image is presented in Figure 3.16a. The output images of the LoG and SoG spatial filters implemented by the proposed circuit using the corresponding kernels are presented in Figure 3.20b and 3.20d, respectively.

Table 3.2 compares the MSE, PSNR, and SSIM performance metrics of spatial

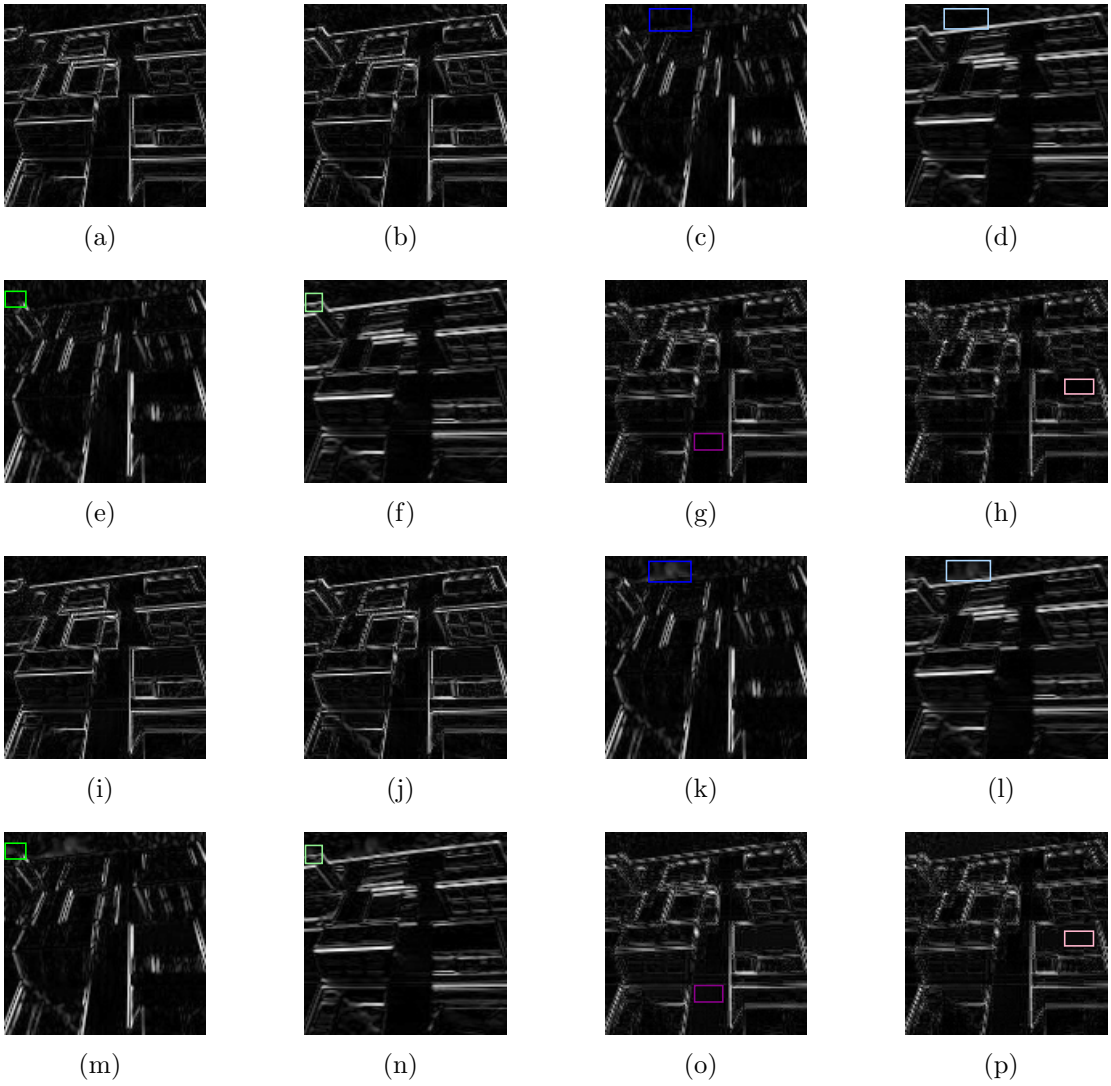


Figure 3.18: Spatial filter circuit for edge detection using 3×3 single kernel: (**a**) software-based x-direction Roberts filter, (**b**) software-based y-direction Roberts filter, (**c**) software-based x-direction Prewitt filter, (**d**) software-based y-direction Prewitt filter, (**e**) software-based x-direction Sobel filter, (**f**) software-based y-direction Sobel filter, (**g**) software-based Laplacian filter,(**h**) software-based modified Laplacian filter, (**i**) hardware-based x-direction Roberts filter, (**j**) hardware-based y-direction Roberts filter, (**k**) hardware-based x-direction Prewitt filter, (**l**) hardware-based y-direction Prewitt filter, (**m**) hardware-based x-direction Sobel filter, (**n**) hardware-based y-direction Sobel filter, (**o**) hardware-based Laplacian filter, and (**p**) hardware-based modified Laplacian filter.

filters for edge detection using the same dataset as Table 3.1. We use post-layout

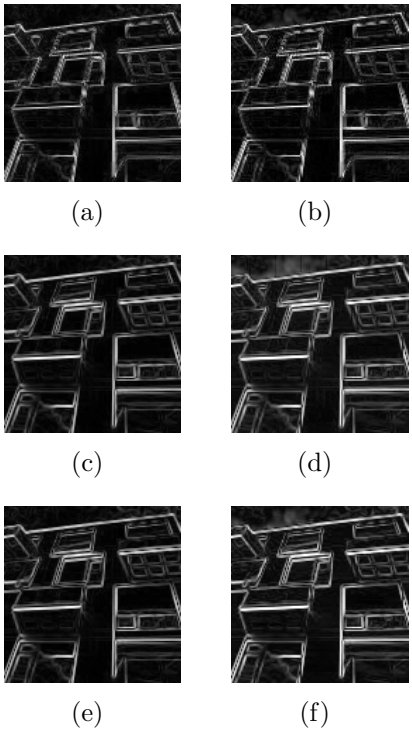
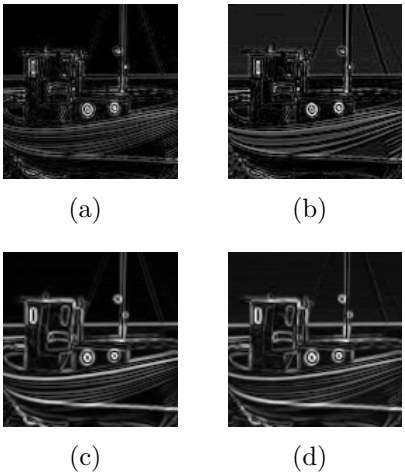


Figure 3.19: Spatial filter circuit for edge detection using two 3×3 kernel on the original image in Figure 3.18a: (**a**) software-based Roberts, (**b**) software-based Prewitt, (**c**) software-based Sobel, (**d**) hardware-based Roberts, (**e**) hardware-based Prewitt, and (**f**) hardware-based Sobel.

Spice simulations to validate the proposed circuit performance. The filters included x-direction Roberts, y-direction Roberts, Roberts, x-direction Prewitt, y-direction Prewitt, Prewitt, x-direction Sobel, y-direction Sobel, Sobel, Laplacian, modified Laplacian, LoG, and SoG. The reference images were obtained using software, while the test images were obtained using hardware. The single-kernel spatial filters, including x-direction Roberts, y-direction Roberts, x-direction Prewitt, y-direction Prewitt, x-direction Sobel, and y-direction Sobel, did not outperform [19, 73, 76] in terms of MSE and PSNR, but they produced similar results. The y-direction Sobel filter exhibited the worst case, with an MSE of 6.27 and a PSNR of 40.15 dB. According to the SSIM metric, the results are excellent and similar to [19, 73, 76]. While the Roberts, Prewitt, and Sobel filters maintain the excellent SSIM performance of the previous group, their MSE and PSNR metrics slightly worsen. The worst case is shown by Sobel with 9.81 and 38.21 dB of MSE and PSNR, respectively. Finally, the LoG and SoG filters show a more significant drop in SSIM, with the SoG value decreasing by 0.11 compared to [73] as the worst case. According to the MSE and PSNR metrics, the LoG filter achieved the best



(a) (b)

(c) (d)

Figure 3.20: Spatial filter circuit for edge detection using two cascaded convolutions with 3×3 different kernels on the original image in Figure 3.16b: (**a**) software-based LoG filter, (**b**) hardware-based LoG filter, (**c**) software-based SoG filter, and (**d**) hardware-based SoG filter.

result among our proposed filters, with values of 4.04 and 42.06 dB, respectively. The SoG filter exhibits MSE and PSNR metrics comparable to the y-direction Sobel filter. Although these results fall short of the work presented in [19, 73, 76] regarding MSE, PSNR, and SSIM metrics, they are still considered good in image processing and computer vision [114, 117]. The results are primarily influenced by the charge injection received in the integration capacitor during integration in both directions. Charge injection errors have a more pronounced impact when employing two cascaded convolutions with different kernels.

Also, we use the Pratt Figure of Merit (PFoM) metric [118] to evaluate the performance of a kernel-based spatial filter configured for edge detection. The PFoM metric measures the accuracy of an edge detector in identifying and localizing true edges in an image, as shown in the following equation:

$$PFoM = \frac{1}{max\left(NX_R, NX_0\right)} \sum_{i=1}^{NX_R} \frac{1}{1 + ad^2\left(i\right)}, \tag{3.17}$$

where NX_R and NX_0 are the number of pixels of the reference and tested image, respectively, d is the distance of separation of an tested edge point to a line of reference edge points and a is the scaling constant (=1/9 following Pratt's original work).

Table 3.2: Comparison of MSE, PSNR and SSIM performance metrics of edge detection filters.

Filter	Resolution	MSE	PSNR	SSIM
[73]	500×500	0.71	49.65	0.99
[74]	100×100	16.53	35.95	0.86
[71]	512×512	93.99	28.40	0.85
[70]	356×533	-	-	0.82
[76]	16×16	2.59	44.00	-
[19]	128×128	0.38	52.3	-
x-direction Roberts*	128×128	4.74	41.57	0.97
y-direction Roberts*	128×128	4.68	41.42	0.97
Roberts*	128×128	7.18	39.57	0.95
x-direction Prewitt*	128×128	5.80	40.49	0.96
y-direction Prewitt*	128×128	6.09	40.28	0.96
Prewitt*	128×128	9.42	38.39	0.94
x-direction Sobel*	128×128	5.84	40.46	0.96
y-direction Sobel*	128×128	6.27	40.15	0.96
Sobel*	128×128	9.81	38.21	0.94
Laplacian*	128×128	5.13	41.02	0.95
modified Laplacian*	128×128	5.18	40.98	0.94
LoG*	128×1288	4.04	42.06	0.89
SoG*	128×128	9.91	38.17	0.88

*Proposed.

Table 3.3: Comparison of PFoM metric of edge detection filters.

Filter	PFoM (%)
[24]	68.94
[38]	95.38
[72]	98.89
Roberts*	95.97
Prewitt*	99.92
Sobel*	99.94
Laplacian*	95.59
modified Laplacian*	93.15

*Proposed.

Table 3.3 compares the proposed circuit and some edge detection filters proposed in the literature to verify the effectiveness of edge detection using the PFoM. We chose a reference edge image obtained by the software-based Sobel because it is most suitable for diagonal edge detection and less sensitive to noise [119]. The Prewitt and Sobel filters give the best results, while the Roberts, Laplacian, and modified Laplacian filters reduce the PFoM by 6.79 %. The increased sensitivity of the last three filters to small changes in intensity and their susceptibility to noise leads to the detection of false edges that decrease the value of the PFoM metric.

4 A CMOS Image Sensor with on-chip defective pixel detection and correction

4.1 Introduction

In this chapter, we outline the proposed algorithm for detecting defective pixels. We detail the CMOS implementation of this algorithm in the analog domain at the pixel level and implement a median filter for correcting the identified defective pixels at the column level, resulting in the SIS. Additionally, we present the results obtained from post-layout simulations, which evaluate the performance of the designed SIS.

4.2 Method

The circuits in our SIS implement detection and correction algorithms for defective pixels. We detect defective pixels by comparing each pixel value to its four neighbors (north, south, east, and west). The correction algorithm replaces the defective pixel value with the median of the same neighbors. In the rest of this section, we describe our proposed pixel detection algorithm.

Our proposed algorithm for detecting defective pixels modifies the algorithm presented in [41] to make it more suitable for implementation on analog hardware. The original algorithm operates within a single-image frame, first defining a 3×3-pixel window centered on the pixel under test, denoted as P_0, as shown in Figure 3.1.

If P_0 has neither the smallest nor the largest value in the window, the algorithm assumes that the pixel is not defective and continues with the next pixel in the image. Otherwise, P_0 is marked as possibly defective, and the algorithm computes an estimate for its value as the average of its eight neighbors, as described in

Equation (4.1):

$$P_{est} = \frac{1}{8}\sum_{i=1}^{8} P_i, \tag{4.1}$$

where P_{est} is the estimated value for P_0 and $P_1 - P_8$ are its neighboring pixels in the 3×3-pixel window. Then, the algorithm computes the average and difference between P_0 and P_{est}, as described in Equations (4.2) and (4.3):

$$P_{avg} = \frac{P_{est} + P_0}{2} \tag{4.2}$$

$$P_{diff} = |P_{est} - P_0|\,. \tag{4.3}$$

The algorithm marks P_0 as a defective pixel when P_{diff} is greater than the absolute difference between P_{avg} and the minimum (0) or maximum (P_{max}) pixel value, as described in Equations (4.4) and (4.5):

$$P_{diff} > P_{avg} \tag{4.4}$$

$$P_{diff} > P_{max} - P_{avg}, \tag{4.5}$$

where P_{max} is the maximum possible value of any pixel in the image.

Assuming that $P_0 < P_{est}$ and substituting Equations (4.2) and (4.3) into Equations (4.4) and (4.5), the algorithm marks P_0 as a dead pixel when one of the conditions described in Equations (4.6) and (4.7) are met:

$$P_0 < \frac{1}{3}P_{est} \tag{4.6}$$

$$P_0 < 3(P_{est} - \frac{2}{3}P_{max}). \tag{4.7}$$

Likewise, when $P_0 \geq P_{est}$, substituting Equations (4.2) and (4.3) into Equations (4.4) and (4.5), the algorithm marks P_0 as a hot pixel when it meets one of the conditions in Equations (4.8) and (4.9):

$$P_0 > 3P_{est} \tag{4.8}$$

$$P_0 > \frac{1}{3}(P_{est} + 2P_{max}). \tag{4.9}$$

In order to simplify the hardware implementation of the algorithm, we propose the following changes to the original algorithm:

1. We use only four pixels in the 3×3-pixel neighborhood to compute the estimated value of P_0, as follows:

$$P_{est} = (P_2 + P_5 + P_7 + P_4)/4. \tag{4.10}$$

2. Instead of using computation to determine whether P_0 is the pixel with the maximum or minimum value in the neighborhood, we mark P_0 as possibly dead when $P_0 < P_L$ and possibly hot when $P_0 > P_H$, with $P_L = 0.05P_{max}$ and $P_H = 0.95P_{max}$ [120, 89, 121].

3. As shown in Figure 4.1a, when $P_{est} < 0.05P_{max}$ and Equation (4.6) is true, then Equation (4.7) is always true. Therefore, we only use Equation (4.6) to detect dead pixels.

4. Likewise, Figure 4.1b shows that when $P_{est} > 0.95P_{max}$ and Equation (4.9) is true, Equation (4.8) is always true. Therefore, we only use Equation (4.9) to detect hot pixels.

5. To compensate for any offset in the analog-circuit implementation of Equations (4.6) and (4.9), we introduce two adjustable thresholds, P_{TL} and P_{TH}. We determine the value of these offsets in Section 4.4. Thus, we rewrite Equations (4.6) and (4.9) as Equations (4.11) and (4.12) for dead and hot pixels, respectively:

$$P_0 - \frac{1}{3}P_{est} < P_{TL} \tag{4.11}$$

$$P_0 - \frac{1}{3}P_{est} - \frac{2}{3}P_{max} > P_{TH}. \tag{4.12}$$

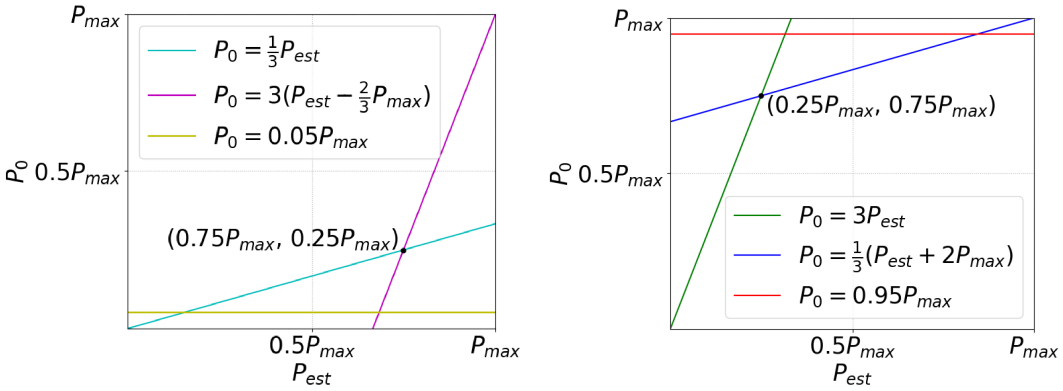


Figure 4.1: Detecting dead and hot pixels using Equations (4.6)–(4.9): (**a**) when $P_{est} < 0.05P_{max}$ and Equation (4.6) is true, Equation (4.7) is true; (**b**) when $P_{est} > 0.95P_{max}$ and Equation (4.9) is true, Equation (4.8) is true.

In summary, Algorithm 1 shows our proposed defective pixel detection algorithm. The next section describes the analog CMOS implementation of our detection and correction algorithm.

Algorithm 1 Proposed detection method.

Input: P_0, P_2, P_4, P_5, P_7, P_L, P_H, P_{TL}, P_{TH}

Output: Boolean value that indicates whether the pixel is non-defective (true) or defective (false)

```
begin
if P_L < P_0 < P_H then
  return false;
else
  P_est ← (P_2 + P_4 + P_5 + P_7)/4;
  if P_TL < P_0 − (1/3)P_est < P_TH + (2/3)P_max then
    return true;
  else
    return false;
  end if
end if
```

4.3 SIS architecture

In the proposed readout circuit, the defective pixel detection stage of the algorithm is implemented in the analog domain at the pixel level. The defective pixel correction stage operates at the column level, again in the analog domain. This allows us to reduce the number of transistors used to implement the algorithm, thereby increasing the fill factor of the imager and reducing its power consumption.

4.3.1 Defective pixel detection circuit

Each pixel in the SIS is composed of a photodetector and a defective pixel detection circuit. This circuit receives as input the current from its photodetector PD0 and its four neighbors, PD2, PD5, PD7, and PD4, as shown in Figure 4.2, and produces a single-bit output that signals whether the pixel is defective or not. The circuit is based on a CTIA, which integrates a photodetector current and produces a voltage representing the intensity of the incident light at the pixel during the integration time. Compared to other readout circuits, the CTIA has a larger dynamic range and higher linearity due to the high gain of its OTA [100, 122]. We used a conventional two-stage OTA [101, 123] for the CTIA and the two comparators.

The circuit shown in Figure 4.2 first compares the value of the current pixel P_0 to the limits P_L and P_H, as required by Line Four of Algorithm 1, then evaluates Equations (4.11) and (4.12). The circuit output V_p is a logic 0 when P_0 is a defective pixel.

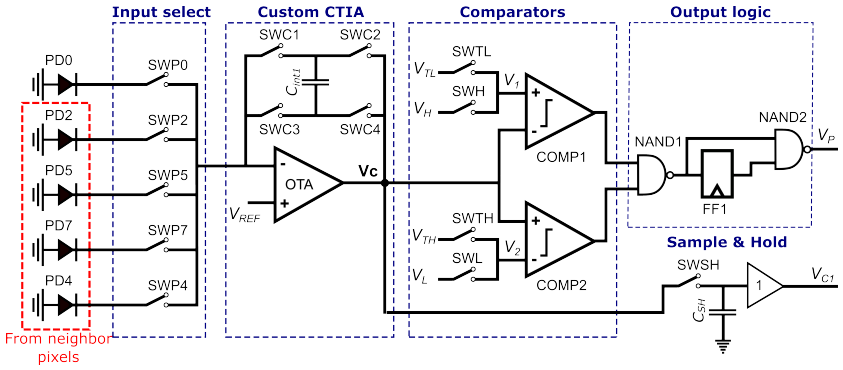


Figure 4.2: Defective pixel detection circuit.

During the integration time T, the circuit operates in two phases. Figure 4.3 shows the activation of the switches SWP0, SWP2, SWP5, SWP7, and SWP4 that select the input to the defective pixel detection circuit of Figure 4.2 during the integration time. Figure 4.3 shows that Phase 1 operates during $T_1 = 3T/4$ and Phase 2 during $T_2 = T/4$. Figure 3.2a shows the equivalent circuit of the defective pixel detector during Phase 1. Switch SWP0 in the Input Select block of Figure 4.2 is closed, and the other input select switches are open, providing the output current of photodetector PD0 as the input to the Custom CTIA block. In this block, switches SWC1 and SWC4 are closed; thus, the CTIA integrates the current generated by the photodetector PD0 in the positive direction. At the end of Phase 1, Equation (4.13) provides the output voltage of the CTIA:

$$V_C = \frac{T_1}{C_{int1}} I_0 + V_{REF}, \tag{4.13}$$

where C_{int1} is the integration capacitor value, I_0 is the PD0 photodetector current, and V_C is the output voltage at the end of Phase 1. Choosing $V_{REF} = V_{dd}/2$ guarantees that the OTA output signal is symmetrical and has a maximum dynamic range. In the Comparators block, switches SWH and SWL are closed; thus, after T_1, the OTA output V_C is compared against V_L and V_H using comparators COMP1 and COMP2, storing a logic 1 in flip-flop *FF1* if either $V_C < V_L$ or $V_C > V_H$ (Line Four of Algorithm 1). At the end of Phase 1, V_C is stored in the Sample & Hold circuit shown at the bottom right corner of Figure 4.2 for later use by the defective pixel correction circuit.

Figure 3.2b shows the equivalent circuit during Phase 2. At the start of this phase, the voltage across the integration capacitor V_C represents P_0. At this point, switches SWC1 and SWC4 are opened, and switches SWC2 and SWC3 are closed, inverting the polarity of the output. The circuit divides Phase 2 into four equal time intervals with duration $T_2/4$. As shown in Figure 4.3, within each

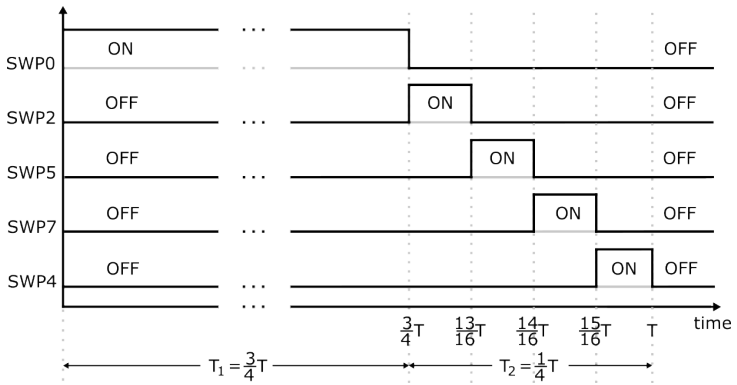


Figure 4.3: Input-select switch states for defective pixel detection during circuit operation Phases 1 and 2. For clarity, the output logic stage is omitted.

time interval, only one of switches SWP2, SWP5, SWP7, or SWP4 is closed at a time; thus, the CTIA successively integrates the current from photodetectors PD2, PD5, PD7, or PD4 in the negative direction. At the end of Phase 2, the output voltage V_C is provided by Equation (4.14):

$$V_C = \frac{T_2}{4C_{int1}}(I_2 + I_5 + I_7 + I_4) - \frac{T_1}{C_{int1}}I_0 + V_{REF} \tag{4.14}$$

where I_2, I_5, I_7, and I_4 are the output currents of the photodetectors PD2, PD5, PD7, and PD4, respectively.

At this point, voltage V_C represents $\frac{1}{3}P_{est} - P_0$. Therefore, instead of evaluating Equations (4.11) and (4.12), the Comparators block evaluates

$$\frac{1}{3}P_{est} - P_0 \;>\; -P_{TL} \tag{4.15}$$

$$\frac{1}{3}P_{est} - P_0 \;<\; -\left(P_{TH} + \frac{2}{3}P_{max}\right), \tag{4.16}$$

by closing switches SWTH and SWTL and opening SWH and SWL. In Figure 4.2, the voltages V_{TL} and V_{TH} represent $-P_{TL}$ and $-(P_{TH} + \frac{2}{3}P_{max})$, respectively. The output of comparators COMP1 and COMP2 is a logic 0 when the corresponding conditions in Equations (4.15) and (4.16) are met, producing a logic 1 at the output of gate NAND1.

Finally, the NAND2 gate in Figure 4.2 produces the output V_P of the defective pixel detection circuit, such that $V_P = 0$ (which signals a defective pixel) when $P_0 < P_L$ or $P_0 > P_H$ and one of the conditions represented by Equations (4.15) and (4.16) (or equivalently, Equations (4.11) and (4.12)) are met.

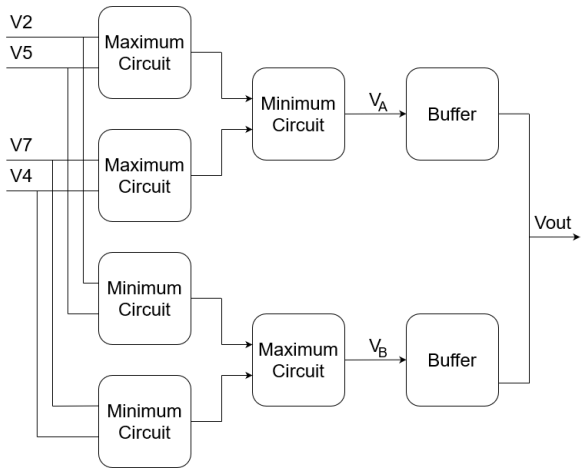


Figure 4.4: Block diagram of defective pixel correction circuit.

4.3.2 Defective pixel correction circuit

Figure 4.4 shows the block diagram of the defective pixel correction circuit. The inputs $V2$, $V5$, $V7$, and $V4$ are the output voltages V_c of the configurable CTIA stage (see Figure 4.2) of the four neighboring pixels. The circuit is a median filter that computes the median value of its four inputs using a two-stage analog sorting network, followed by an output stage that computes the average value of the two central inputs, V_A and V_B, after sorting:

$$V_{out} = \frac{V_A + V_B}{2}, \tag{4.17}$$

where

$$V_A = \min\left[\max\left(V2, V5\right), \max\left(V7, V4\right)\right] \tag{4.18}$$

$$V_B = \max\left[\min\left(V2, V5\right), \min\left(V7, V4\right)\right]. \tag{4.19}$$

The buffers are implemented using a single-stage OTA [101, 123], and the sorting network comprises circuits that compute the minimum and maximum value of their two inputs. Figures 4.5a and 4.5b show the minimum and maximum circuits, respectively.

Figure 4.5 shows the circuits that compute the minimum and maximum operations used in Equations (4.18) and (4.19), which were proposed by Soleimani [124]. Both circuits consist of a differential amplifier stage composed of M_{I1}, M_{I2}, M_{out}, M_{C1}, M_{C2}, and M_{Cout}, along with a common-source stage with an active load, composed of M_{P1}, M_{P2}, and M_2. Because the operation of both circuits is similar, we explain only the operation of the minimum-value circuit shown in Figure 4.5a. When $V_{in1} < V_{in2}$, M_{I2} is almost off, the drain currents of M_{I1} and M_{out} are equal

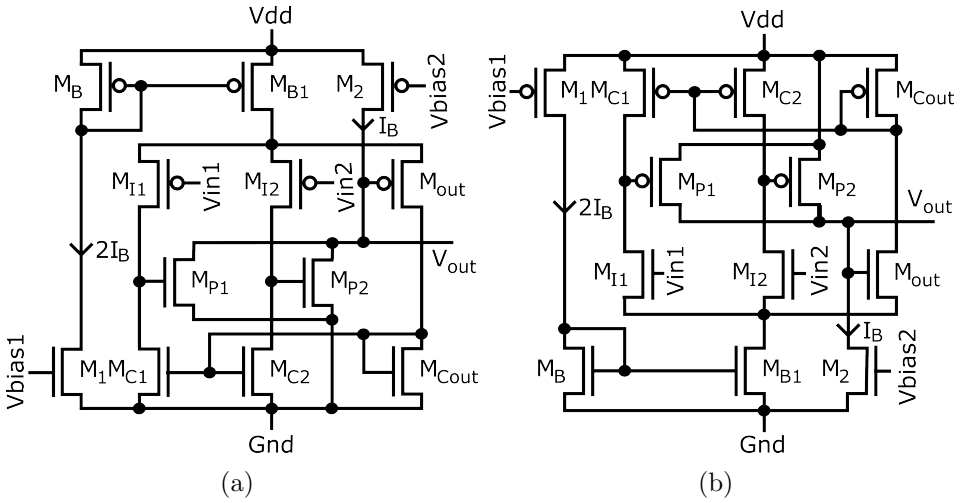


Figure 4.5: Circuit diagram to determine the minimum and maximum values between two voltages.

to I_B because of the current mirror formed by M_{C1} and M_{Cout}. Due to the state of M_{I2}, the gate voltage of M_{P2} is very small, and this transistor is almost turned off as well. Therefore, all of the current I_B from M_2 passes through M_{P1}. Because the current passing through M_{P1} and M_{Cout} has the same value I_B, the gate voltages V_{GS} of both transistors are the same. Finally, the source and drain voltages and drain current of M_{I1} and M_{out} are equal, which results in $V_{out} = V_{in1}$. The same reasoning applies when $V_{in2} < V_{in1}$, in which case $V_{out} = V_{in2}$. The circuit operation in Figure 4.5b follows the same principle, except with the computation of the maximum value between the two inputs, V_{in1} and V_{in2}.

4.3.3 General readout circuit

Figure 4.6 shows the 128×128 smart pixel array. Each smart pixel, denoted as P&DC, includes a photodetector and a defective pixel detection circuit. Each column includes a defective pixel correction circuit (CC) and a 2:1 analog multiplexer. An External digital device generates the timing and control signals to the array, and the 128:1 analog multiplexer generates the output.

Each P&DC circuit has two outputs: the output value V_C of the defective-pixel detection circuit at the end of Phase 1 and the output value V_P indicating whether the pixel is defective. Each column reads the V_C output of three smart pixels, which are the center P_0, top P_2, and bottom P_7 pixels of the 3×3-pixel window in Figure 3.1, as well as the V_P output of the central pixel P_0. The pixel values V_C are stored in a sample and hold circuit at the end of Phase 1. The defective pixel correction circuit uses the P_2 and P_7 outputs along with the P_4 and

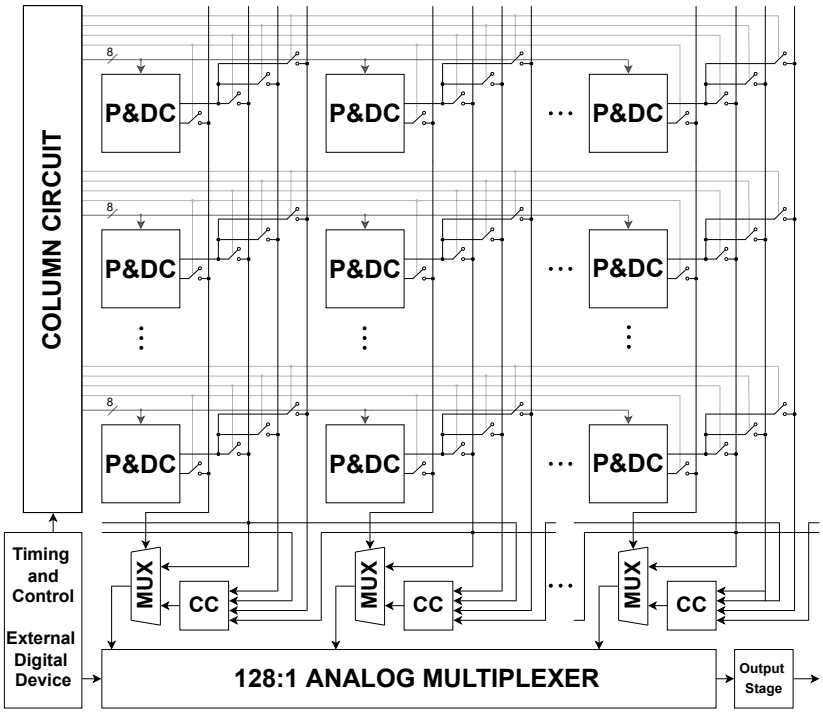


Figure 4.6: Smart pixel array.

P_5 outputs of the neighboring columns to compute the corrected pixel value V_{out}. The 2:1 multiplexer selects between P_0 and the corrected value V_{out} depending on the logic value of V_P, which indicates whether or not P_0 is defective. The 128:1 analog multiplexer serially outputs all the pixel voltage values within each row.

4.4 Results

4.4.1 Physical layout

The EDA tools used in this work are the same as those employed in Section 3.4.1. Figure 4.7 shows the physical layout of the defective pixel detection circuit on a 0.35 μm mixed-signal CMOS processor with three metal layers, two polysilicon layers, and a 3.3 V supply voltage.

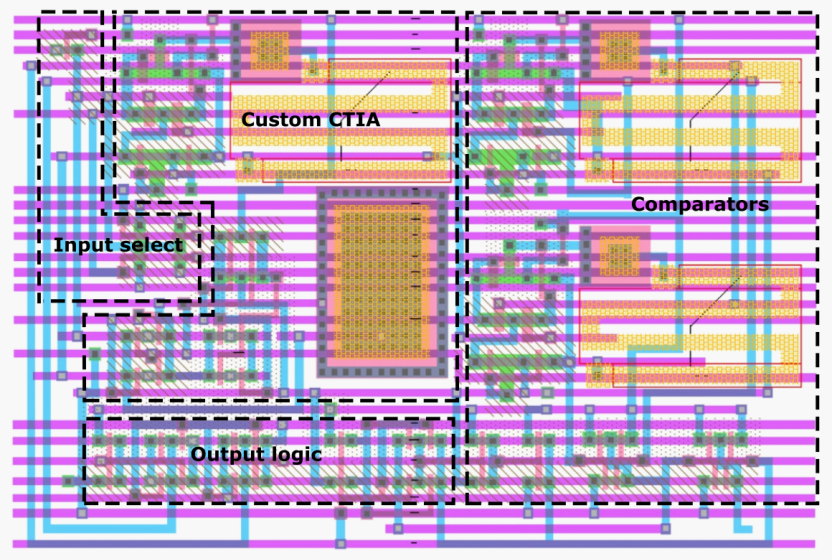


Figure 4.7: Layout of the defective pixel detection circuit.

We used a double-poly capacitor with a capacitance per unit area of 950 aF/μm^2. Assuming an integration time $T = 16$ μs (Section 4.3.1) and a maximum photodetector current of 14 nA, the pixel requires a 100 fF integration capacitor of 13.1 μm by 8.0 μm. The area of the complete defective pixel detection circuit is 42 μm by 63 μm. Considering a 64 μm by 64 μm pixel [125], the circuit achieves a fill factor of 35.4%, which is in the range of typical values for the technology used, as shown in Section 3.4.1. In comparison, the fill factor without these additional stages is 73.4%. It is worth noting that only two wires are required between neighboring cells, each transporting photocurrent in a different direction. This is because, as described in Section 4.3.1, the defective pixel detection circuit implements Equations (4.10)–(4.12) using time multiplexing, reading the photocurrent of only one pixel at a time. Moreover, because, at most, only one of the two wires operates in low impedance simultaneously, the effects of crosstalk are diminished.

Although the results presented in this section use the 0.35 μm physical design described above, we additionally evaluated the scalability of our circuits by porting the 0.35 μm physical design to a standard 0.18 μm CMOS processor frequently used in the literature [126, 127, 128, 129]. The 0.18 μm processor uses a 1.8 V supply voltage and features MIM capacitors with a capacitance per unit area of 2 fF/μm^2. The total area of the defective pixel detection circuit in the 0.18 μm process is 30.1 μm by 45.2 μm, which achieves a fill factor of 66.7% in the same 64 μm by 64 μm pixel. Without the comparators and output logic stages, the fill factor is 86.3%.

Figure 4.8 shows the layout of the defective pixel correction circuit, which has

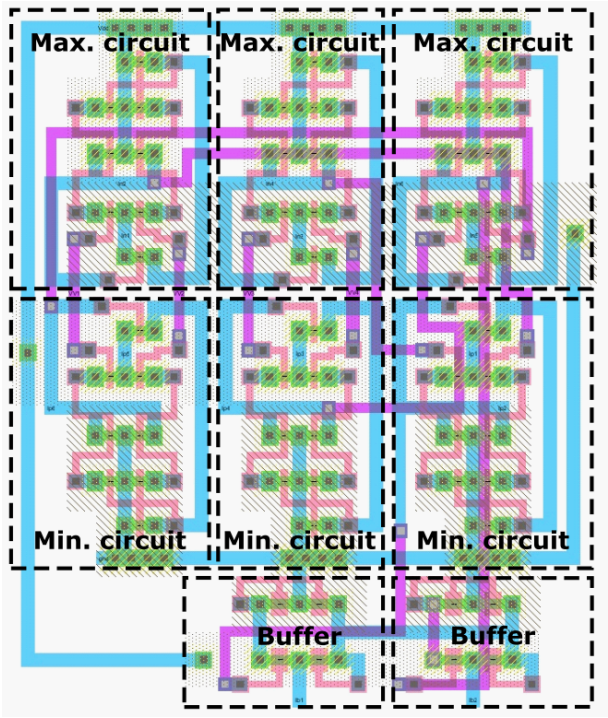


Figure 4.8: Layout of defective pixel correction circuit.

an area of 42 μm by 35 μm in the 0.35 μm standard CMOS process and 21.2 μm by 17.8 μm in the 0.18 μm standard CMOS process.

4.4.2 Algorithm parameters

To select the values of P_{TL} and P_{TH} in Algorithm 1, we used 250 images with different levels of brightness, contrast, and dynamic range randomly selected from the Linnaeus 5 dataset [1]. To determine the value for P_{TL}, we randomly added 0.5% dead pixels to the image with values between 0 and $0.05P_{max}$. Similarly, we added 0.5% hot pixels (between $0.95P_{max}$ and P_{max}) to determine the value of P_{TH}.

We simulated the performance of the algorithm as a function of the values of P_{TH} and P_{TL} in the interval $[-0.025P_{max}, 0.025P_{max}]$. We evaluated the performance of the algorithm using four metrics: Sensitivity (Se), Specificity (Sp), precision or PPV, and Phi coefficient (ϕ), defined as follows:

$$Se = \frac{TP}{TP + FN} \tag{4.20}$$

$$Sp = \frac{TN}{TN + FP} \tag{4.21}$$

$$PPV = \frac{TP}{TP + FP} \tag{4.22}$$

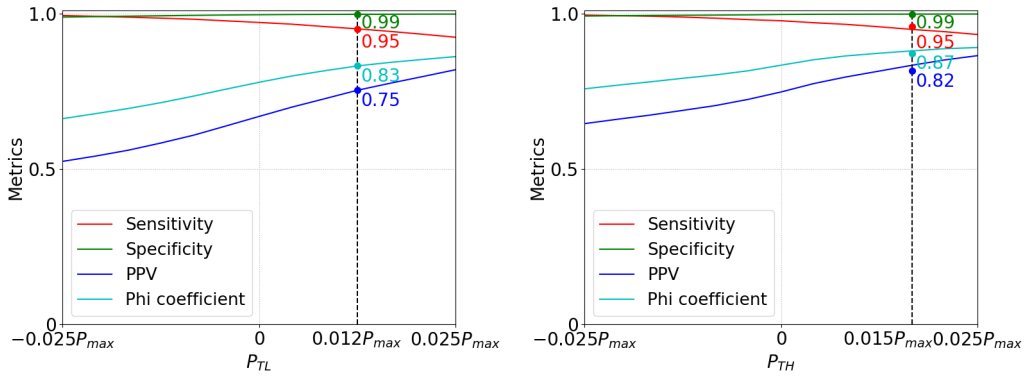
$$\phi = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}, \tag{4.23}$$

where True Positives (TP) is the number of true positive defective pixels in the image detected by the algorithm, True Negatives (TN) is the number of true negatives, False Positives (FP) is the number of false positives, and False Negatives (FN) is the number of false negatives; Sensitivity quantifies the fraction of pixels marked as defective by the algorithm that are truly defective; Specificity expresses the fraction of good pixels correctly identified as such by the algorithm; PPV is the precision of the algorithm, that is, the fraction of truly defective pixels that were detected; and the ϕ coefficient is a special case of the Pearson correlation coefficient that measures the difference between the ratios of correctly and incorrectly classified pixels.

Figures 4.9a and 4.9b show the four metrics averaged over the 250 images for a range of P_{TL} and P_{TH} values. Figure 4.9a shows the performance of the algorithm when detecting dead pixels, and Figure 4.9b depicts the performance metrics when detecting hot pixels. These graphs show that increasing the value of P_{TL} and P_{TH} improves the PPV and ϕ metrics and degrades Se, while the Specificity remains approximately constant at $Sp = 0.99$. Based on these results, we chose $P_{TL} = 0.012P_{max}$ and $P_{TH} = 0.015P_{max}$, which resulted in a sensitivity of $Se = 0.95$. With these values, we achieve $\phi = 0.83$, $PPV = 0.75$ when detecting dead pixels, $\phi = 0.87$, and $PPV = 0.82$ when detecting hot pixels.

4.4.3 Post-layout simulation

We randomly selected 500 images with different levels of brightness, contrast, and DR from the Linnaeus 5 dataset [1] to create the images used to evaluate the detection and correction algorithms. This set of test images is different from the set of training images used to obtain the values of P_{TL} and P_{TH}. All images have the same 128×128 resolution size, matching the proposed CMOS readout circuit array size. The most common type of defective pixel is the random distribution of dead and hot pixels in an image. In this paper, we use two variants of this type of defective pixel: 0.5% randomness (0.5% of all pixels in the image are defective) and 1.0% randomness (1.0% of all pixels in the image are defective). Furthermore,



(a) (b)

Figure 4.9: Performance of the algorithm using Sensitivity, Specificity, PPV, and Phi coefficient as functions of P_{TL} and P_{TH}.

we impose the requirement that the dead pixel and hot pixel ranges be the lowest and highest 5% of the full DR, respectively. These values are consistent with the evaluation of other algorithms reported in the literature [88, 89, 93, 41, 83].

Figure 4.10 shows a post-layout simulation of the defective pixel detection circuit. The graphs in the figure show the voltages at four nodes of the circuit shown in Figure 4.2: the output V_C of configurable CTIA, the reference inputs V_1 and V_2 of the comparators COMP1 and COMP2, and the output V_P, which is a logic 1 when the pixel is not defective. Figure 4.10a shows an example where the input pixel is not defective and passes the test in Line Four of Algorithm 1 ($P_L < P_0 < P_H$). At $t = 0$, FF1 is preset to 1, and capacitor C_{int} is discharged. Between $t = 0$ and $t = 12$ μs, the circuit operates in Phase 1, integrating the value of the input pixel P_0, and the switches at the input of COMP1 and COMP2 select $V_1 = V_H$ and $V_2 = V_L$ as reference voltages. At $t = 12$ μs, both comparators output a logic 1 because $P_H > P_0$ and $P_0 > P_L$; thus, FF1 stores a logic 0 and $V_P = 1$. At the same time, switch $SWSH$ is opened to sample the pixel value. At $t = 12.25$ μs, the CTIA changes its integration direction, and the circuit initiates Phase 2. Switch SWP0 opens and, for the next four 1 μs-intervals, switches SWP2, SWP5, SWP7, and SWP4 select the neighboring pixels as inputs to the CTIA. At $t = 14$ μs, the input switches of COMP1 and COMP2 select $V_1 = V_{TL}$ and $V_2 = V_{TH}$. At $t = 16.25$ μs, Phase 2 ends, and COMP1 and COMP2 output a logic 1 (the input pixel P_0 passes the test in Line Eight of Algorithm 1), and $V_P = 1$.

Note that in Figure 4.10a, it is irrelevant whether or not P_0 passes the test in Line Eight because it has already passed the test in Line Four. In contrast, Figure 4.10b shows a case where the P_0 does not pass the test in Line Four of

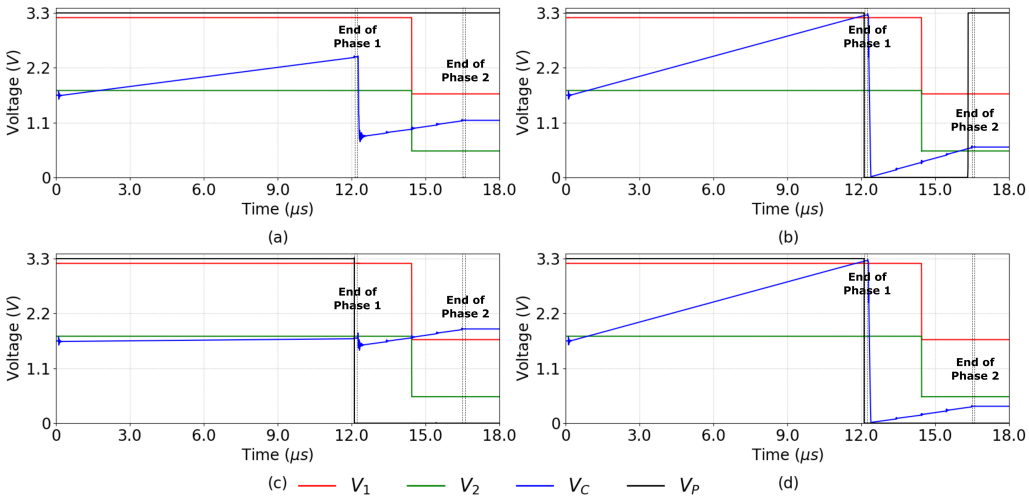


Figure 4.10: Post-layout simulation of defective pixel detection circuit: (**a**) a good pixel that passes the test in Line Four of Algorithm 1; (**b**) a good pixel that fails the test in Line Four of Algorithm 1; (**c**) a dead pixel; (**d**) a hot pixel.

Algorithm 1 because $P_0 > P_H$. Nevertheless, the pixel is not defective because it does pass the test in Line Eight. At the end of Phase 1, COMP1 outputs a logic 0, and COMP2 outputs a 1; therefore, FF1 stores a logic 1 (marking the pixel as potentially defective) and $V_P = 0$. However, at the end of Phase 2, both comparators output a logic 1 because the pixel passes the test in Line Eight; thus, the NAND1 in Figure 4.2 outputs a logic 0 and $V_P = 1$, marking the pixel as not defective.

Figures 4.10c and 4.10d present examples where P_0 does not pass the test in Line Four of Algorithm 1 because $P_0 < P_L$ or $P_0 > P_H$, respectively. Furthermore, in both examples, P_0 does not pass the test in Line Eight because $P_0 - \frac{1}{3}P_{est} < P_{TL}$ or $P_0 - \frac{1}{3}P_{est} > P_{TH} + \frac{2}{3}P_{max}$, respectively.

In the case illustrated in Figure 4.10c, at the end of Phase 1 COMP1 outputs a logic 1 and COMP2 outputs a 0 ($P_0 < P_L$). Therefore, FF1 stores a logic 1 and $V_P = 0$. At the end of Phase 2, COMP1 outputs a 0 because $P_0 - \frac{1}{3}P_{est} < P_{TL}$, while COMP2 outputs a 1. Thus, the first NAND in Fig 4.2 outputs a logic 1 and $V_P = 0$, marking the pixel as defective. In the case shown in Figure 4.10d, at the end of Phase 1, COMP1 outputs a logic 0 ($P_0 > P_H$), and COMP2 outputs a 1; thus, FF1 stores a logic 1 and $V_P = 0$. At the end of Phase 2, COMP1 outputs a logic 1 and COMP2 outputs a 0 because $P_0 - \frac{1}{3}P_{est} > P_{TH} + \frac{2}{3}P_{max}$. Thus, the first NAND in Figure 4.2 outputs a logic 1 and $V_P = 0$, marking the pixel as defective.

Figure 4.11 shows a post-layout simulation of the median computation in the

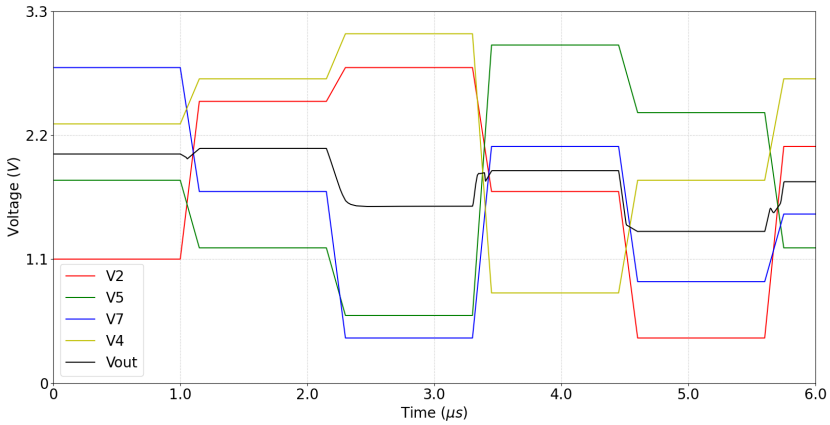


Figure 4.11: Post-layout simulation of the defective pixel correction circuit.

defective pixel correction circuit, as shown in Figure 4.4. In this test, we simulate the readout of five consecutive pixels of a column in the image. For each central pixel P_0, the row-select circuit outputs the voltage values of its four neighbors, denoted as $V2$, $V5$, $V7$, and $V4$ in the figure. For each neighbor, their voltage value is provided by the sample-and-hold circuit shown at the bottom right corner of Figure 4.2. As shown in Figure 4.11, we select a new pixel every 1 μs; after a delay of 150 ns, the output voltage V_{out} settles at the median value shown in Equation 4.17, with a worst-case error of 14.8% in the pixel range. The delay is introduced by the minimum, maximum, and buffer circuits shown in Figure 4.4. The worst-case delay occurs when the minimum or maximum circuit at the input changes its output between two consecutive pixels (e.g., when first $V2 > V5$ and next $V5 > V2$), and that change propagates through to V_{out}. Moreover, the delay is maximal when the output voltage swing is close to V_{dd}. In this case, the worst-case settling time for V_{out} is 800 ns.

Figure 4.10 shows that the defective pixel detection circuit operates in 16.25 μs, plus 50 ns, to reset the integration capacitor after each detection. This circuit operates in parallel for all pixels in the image. During readout, the row-select circuit has a delay of 60 ns, the worst-case delay of the correction circuit is 800 ns at the column level, and the column-select circuit has a delay of 80 ns. In this case, the readout time for the complete array is 1.44 ms, which allows for acquiring and processing images at a maximum frame rate of 694 fps.

4.4.4 Circuit performance

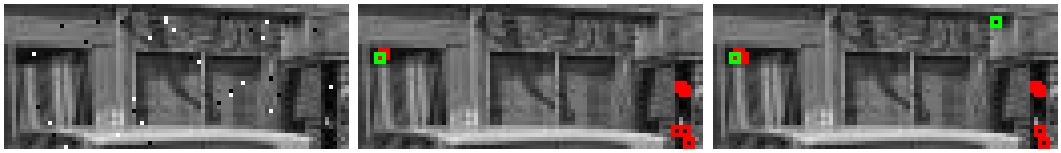
Figure 4.12 illustrates the operation of the proposed algorithm and circuit. Figure 4.12a shows three images taken from the Linnaeus 5 dataset [1], contaminated with salt-and-pepper noise randomly affecting 1% of the pixels. Dead pixels were

(a) (b) (c)

Figure 4.12: Original images taken from the Linnaeus 5 dataset [1] with 1.0 % defective pixels added randomly along with example images showing the results of the defective pixel detection and correction process: (**a**) Original images with defective pixels. (**b**) Results of the proposed algorithm. (**c**) Results of the readout circuit.

generated in a range of $[0, 0.05P_{max}]$, while hot pixels were generated in a range of $[0.95P_{max}, P_{max}]$. Figures 4.12b and 4.12c show the outputs of a software implementation of Algorithm 1 and our detection/correction circuit, respectively. In Figures 4.12b and 4.12c, false positive detections are highlighted in red, and false negatives are marked in yellow.

To produce the images in Figure 4.12c, pixel values in the input image were converted to currents in the range from 0 to 14 nA in order to simulate the operation of a photodetector. Then, we ran a post-layout Spice simulation of the circuits described in Section 4.3 to produce the output pixel values and the list of pixels marked as defective by the circuit.F inally, we converted the output pixel voltages from 0 to 3.3 V to digital values between 0 and 255 to display the output images.



(a) (b) (c)

Figure 4.13: Zoom of a region of the original image taken from tthe Linnaeus 5 dataset [1] with FP and FN examples: (**a**) Noisy image. (**b**) Results of proposed algorithm. (**c**) Results of the proposed circuit.

A visual inspection of Figure 4.12 shows that the algorithm detects and corrects most defective pixels. Moreover, our pixel-correction circuit produces an output that is visually almost identical to the software implementation of the algorithm.

Figure 4.13 shows an enlarged version of the 35×85-pixel rectangle highlighted in cyan in the first image of Figure 4.12. Figures 4.13b and 4.13c show the output of the algorithm and correction circuit, respectively, with false positive detections highlighted in red and false negatives marked in green. Out of the 35 defective pixels in the image, the algorithm correctly detects 34, and the circuit detects 33. Moreover, out of the 2940 good pixels, the algorithm and circuit produce only seven false positives, which differ in only one detection. All the false positives and false negatives in Figure 4.13 correspond to dead pixels, which is explained by the mostly dark background in the image, and these do not significantly affect the appearance of the image.

Figure 4.14 compares the performance of our algorithm and hardware implementation to six other defective pixel detection algorithms, namely, those proposed by Chan [88], Yongji [93], Cho [89], the original algorithm proposed by Tchendjou et al., its simplification [41] (Tchendjou1 and Tchendjou2), and the median-based version of Tchendjou1 [83] (Tchendjou3), which does not requires dataset-dependent parameters. The figure plots the Se, Sp, PPV, and ϕ metrics, defined in Section 4.4.2, averaged over 500 128×128-pixel images randomly selected from the Linnaeus 5 dataset [1], to which we added salt-and-pepper noise affecting 0.5% and 1.0% of the pixels. These images are different from the 250 images used to determine the values of P_{TL} and P_{TH} in Section 4.4.2. The hardware implementation results were obtained from post-layout Spice simulations.

Figure 4.14a shows that Cho achieves the highest sensitivity (96.87%), though all the algorithms except for Yongji achieve a sensitivity above 93%. In Figure 4.14b–d, it can be seen that our algorithm and its hardware implementation achieve the highest performance based on the Sp, PPV, and ϕ metrics.

In summary, the classification performance of our algorithm is competitive with the state-of-the-art algorithms compared in Figure 4.14. It is based on simple

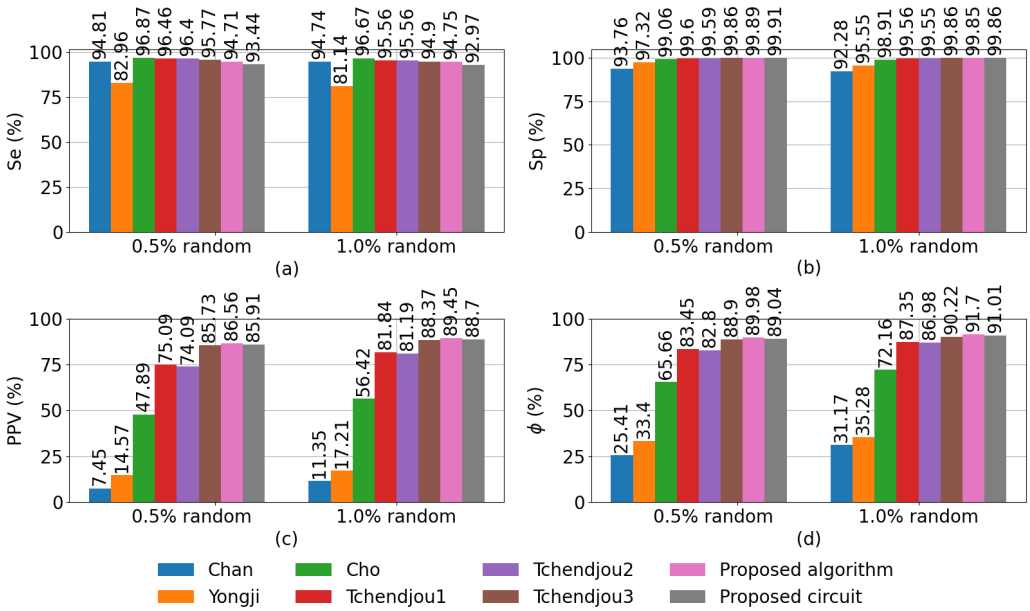


Figure 4.14: Comparison of defective pixel detection methods using (**a**) Sensitivity, (**b**) Specificity, (**c**) PPV, and (**d**) Phi coefficient.

arithmetical operations that can be efficiently implemented in analog hardware with comparable performance to software implementations.

To assess the combined performance of the detection and correction algorithms and their hardware implementations, we used the PSNR and Image Enhancement Factor (IEF) metrics. The PSNR metric compares a reference image I_0 without defective pixels to an image I_R, which results from applying a correction algorithm to a noisy version of I_0. The PSNR was defined by Equation (3.15).

IEF is computed as the ratio between the IEF of the noisy image and the IEF of the corrected image, using the noiseless image as a reference, as shown in Equation (4.24):

$$IEF = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I_N(i,j) - I_0(i,j)\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(I_R(i,j) - I_0(i,j)\right)^2}. \tag{4.24}$$

where I_0 and I_R are the noiseless and corrected images, respectively, and I_N is the noisy image. If the correction algorithm reduces the MSE, then $IEF > 1$; conversely, if the algorithm increases the MSE, then $IEF < 1$.

Figure 4.15 compares the PSNR and IEF of our detection and correction algorithms and their hardware implementations to five algorithms proposed in the literature, labeled in the image as Chan [88], Yongji [93], Cho [89], Tchendjou1 [41], and Tchendjou3 [83]. In all cases, the algorithms performed defective pixel de-

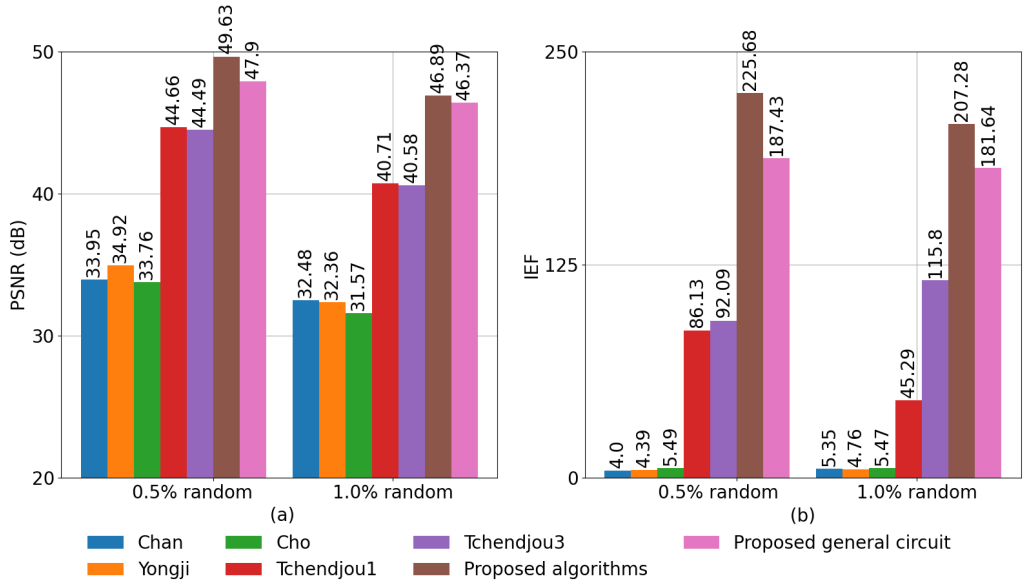


Figure 4.15: Comparison of defective pixel detection/correction methods using (**a**) PSNR, and (**b**) IEF.

tection and correction on the same set of images used to generate the results shown in Figure 4.14. The hardware implementation results were obtained from post-layout Spice simulations of the circuits described in Section 4.3. Figure 4.15 shows that our algorithm performs better than all the other detection/correction methods measured by both metrics. Our PSNR results are 11%–15% higher than Tchendjou1 and Tchendjou3, and are more than 42% higher than the other three methods. In addition, the IEF achieved by our algorithm is more than 79% higher than the value achieved by Tchendjou1 and Tchendjou3, and more than 38 times the IEF achieved by Chan, Yongji, and Cho. These three methods produce a many FP, as shown in the PPV results in Figure 4.14c. The IEF metric is highly sensitive to the number of false positives, resulting in significant degradation in the performance of these three methods. Lastly, it can be observed that the PSNR of the analog hardware implementation is 1%–3.7% worse than the algorithm, and the IEF is 12%–17% lower. It is mainly due to charge injection in the detection circuit and the nonlinearities and offsets in the response of the detection and median computation circuits.

5

Conclusions and future work

This thesis reports the design and evaluation of two SIS of 128 × 128 píxeles using a 0.35 μm CMOS process, one for kernel-based spatial filtering calculation and the other for detecting and correcting defective pixels. The first SIS performs directly at the pixel level and can be configured to perform spatial filtering that requires a single kernel for convolution, such as mean, Gaussian, x- and y-direction Roberts, x- and y-direction Prewitt, x- and y-direction Sobel, Laplacian, and modified Laplacian. It can also perform two convolutions using different kernels on the same input image, such as Roberts, Prewitt, and Sobel, or two cascaded convolutions using different kernels, such as LoG and SoG. Therefore, SIS can be employed for denoising, and edge detection to extract features. The second SIS detects defective pixels (dead and hot pixels) at the pixel level and uses a simplification of the algorithm presented in [41], while the correction is made at the column level and replaces a pixel's value with the median value of its four neighbors.

The kernel-based spatial filtering SIS processes images at frame rates ranging from 743 to 752 frames per second (fps) and achieves a fill factor of 20.6 %, according to post-layout simulation. This SIS demonstrates acceptable performance in both denoising and edge detection tasks. The results are equivalent to a software implementation and other SIS published according to quality image metrics. The second SIS that detects and corrects defective pixels processes images at 694 fps and achieves a fill factor of 35.4 %, according post-layout simulation. In compliance with quality image metrics, this SIS achieves slightly lower performance than the software implementation equivalent, though it exhibits better performance than the software versions of the other published methods. Similar to other detection/correction algorithms based on local statistics, our design is sensitive to the occurrence of two or more defective pixels within a small window.

The results show that when algorithms can be parallelized and are based on

simple arithmetic operations between pixels in a window, their implementation in on-pixel analog hardware (or at column level in the case of pixel correction) increases the processing frame rate and reduces the power of the overall system including the high-performance external device that performs the rest of the processing. However, reducing the effective area available for light capture entails a decreased fill factor, which may be acceptable depending on the application, given the associated benefits. The CTIA in each SIS pixel enables the photocurrent integration in a parallel fashion to perform part of the computation without using extra arithmetic circuits or buffers, increasing the frame rate and reducing the impact on fill factor and data accuracy.

In terms of future work, we are extending our work to other algorithms that can exploit pixel-level parallelism in hardware using the same approach presented in this thesis, thereby enhancing the capabilities of IoT devices to process visual data in real time efficiently. In particular, we focus on the extraction of complex temporal and spatial features from continuous video streams, facilitating tasks such as real-time object tracking and gesture recognition. Additionally, we are focusing on Non-uniformity Correction (NUC) algorithms to autonomously calibrate infrared sensor arrays, compensating for inherent pixel variations and environmental factors. The following integrated circuits will also be designed and fabricated using the Skywater open-source 130 nm CMOS Process Design Kit (PDK).

Bibliography

[1] G. Chaladze and L. Kalatozishvili, "Linnaeus 5 dataset for machine learning," 2017.

[2] M. Maheepala, M. A. Joordens, and A. Z. Kouzani, "Low power processors and image sensors for vision-based iot devices: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1172–1186, 2021.

[3] A. Samuel and C. Sipes, "Making internet of things real," *IEEE Internet of Things Magazine*, vol. 2, no. 1, pp. 10–12, 2019.

[4] M. Haghi Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of iot in healthcare: Applications, techniques, and trends," *Journal of Network and Computer Applications*, vol. 192, p. 103164, 2021.

[5] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, "Internet of things market analysis forecasts, 2020–2030," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 449–453.

[6] A. Aslam and E. Curry, "A survey on object detection for the internet of multimedia things (iomt) using deep learning and event-based middleware: Approaches, challenges, and future directions," *Image and Vision Computing*, vol. 106, p. 104095, 2021.

[7] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Multimedia internet of things: A comprehensive survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020.

[8] K. Cai, X. Wu, X. Liang, and K. Wang, "Hardware design of sensor nodes in the nilaparvata lugens monitoring system based on the internet of things," in *Advanced Electrical and Electronics Engineering: Volume 2*. Springer, 2011, pp. 571–578.

[9] S. A. V. Shajihan, T. Hoang, K. Mechitov, and B. F. Spencer Jr, "Wireless smartvision system for synchronized displacement monitoring of railroad bridges," *Computer-Aided Civil and Infrastructure Engineering*, vol. 37, no. 9, pp. 1070–1088, 2022.

[10] J. Peixoto, J. Sousa, R. Carvalho, G. Santos, J. Mendes, R. Cardoso, and A. Reis, "Development of an analog gauge reading solution based on computer vision and deep learning for an iot application," *Telecom*, vol. 3, no. 4, pp. 564–580, 2022.

[11] A. Hartmannsgruber, J. Seitz, M. Schreier, M. Strauss, N. Balbierer, and A. Hohm, "Cube: A research platform for shared mobility and autonomous driving in urban environments," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2315–2322.

[12] M. El-Desouki, M. Jamal Deen, Q. Fang, L. Liu, F. Tse, and D. Armstrong, "Cmos image sensors for high speed applications," *Sensors*, vol. 9, no. 1, pp. 430–444, 2009.

[13] E. R. Fossum, J. Hynecek, J. Tower, N. Teranishi, J. Nakamura, P. Magnan, and A. J. P. Theuwissen, "Special issue on solid-state image sensors," *IEEE Transactions on Electron Devices*, vol. 56, no. 11, pp. 2376–2379, 2009.

[14] K. Hassanli, S. M. Sayedi, R. Dehghani, A. Jalili, and J. J. Wikner, "A highly sensitive, low-power, and wide dynamic range cmos digital pixel sensor," *Sensors and Actuators A: Physical*, vol. 236, pp. 82–91, 2015.

[15] I. Cevik, X. Huang, H. Yu, M. Yan, and S. U. Ay, "An ultra-low power cmos image sensor with on-chip energy harvesting and power management capability," *Sensors*, vol. 15, no. 3, pp. 5531–5554, 2015.

[16] D. Durini, *High performance silicon imaging: fundamentals and applications of cmos and ccd sensors*. Woodhead Publishing, 2019.

[17] D. Ravì, C. Wong, B. Lo, and G.-Z. Yang, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 56–64, 2017.

[18] R. K. Verma, K. Pattanaik, S. Bharti, and D. Saxena, "In-network context inference in iot sensory environment for efficient network resource utilization," *Journal of Network and Computer Applications*, vol. 130, pp. 89–103, 2019.

[19] T.-H. Hsu, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "A 0.5-v real-time computational cmos image sensor with programmable kernel for feature extraction," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 5, pp. 1588–1596, 2021.

[20] R. Joshi, M. A. Zaman, and S. Katkoori, "Novel bit-sliced near-memory computing based vlsi architecture for fast sobel edge detection in iot edge devices," in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 2020, pp. 291–296.

[21] B. M. López-Portilla, W. Valenzuela, P. Zarkesh-Ha, and M. Figueroa, "A cmos image readout circuit with on-chip defective pixel detection and correction," *Sensors*, vol. 23, no. 2, 2023.

[22] T.-H. Hsu, G.-C. Chen, Y.-R. Chen, C.-C. Lo, R.-S. Liu, M.-F. Chang, K.-T. Tang, and C.-C. Hsieh, "A 0.8v intelligent vision sensor with tiny convolutional neural network and programmable weights using mixed-mode processing-in-sensor technique for image classification," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[23] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," *Nature Electronics*, vol. 3, no. 11, pp. 664–671, 2020.

[24] M.-J. Park and H.-J. Kim, "A real-time edge-detection cmos image sensor for machine vision applications," *IEEE Sensors Journal*, vol. 23, no. 9, pp. 9254–9261, 2023.

[25] G. R. C. Fiorante, J. Ghasemi, P. Zarkesh-Ha, and S. Krishna, "Spatio-temporal bias-tunable readout circuit for on-chip intelligent image processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 11, pp. 1825–1832, 2016.

[26] W. Valenzuela, J. E. Soto, P. Zarkesh-Ha, and M. Figueroa, "Face recognition on a smart image sensor using local gradients," *Sensors*, vol. 21, no. 9, p. 2901, 2021.

[27] W. Valenzuela, A. Saavedra, P. Zarkesh-Ha, and M. Figueroa, "Motion-based object location on a smart image sensor using on-pixel memory," *Sensors*, vol. 22, no. 17, p. 6538, 2022.

[28] M. El-Desouki, M. Jamal Deen, Q. Fang, L. Liu, F. Tse, and D. Armstrong, "Cmos image sensors for high speed applications," *Sensors*, vol. 9, no. 1, pp. 430–444, 2009.

[29] W. Valenzuela, J. E. Soto, P. Zarkesh-Ha, and M. Figueroa, "Face recognition on a smart image sensor using local gradients," *Sensors*, vol. 21, no. 9, 2021.

[30] W. Valenzuela, A. Saavedra, P. Zarkesh-Ha, and M. Figueroa, "Motion-based object location on a smart image sensor using on-pixel memory," *Sensors*, vol. 22, no. 17, 2022.

[31] N. Athreyas, D. Gupta, and J. Gupta, "Analog signal processing solution for machine vision applications," *Journal of Real-Time Image Processing*, vol. 16, pp. 1607–1628, 2019.

[32] V. Alimisis, M. Gourdouparis, C. Dimas, and P. P. Sotiriadis, "A 0.6 v, 3.3 nw, adjustable gaussian circuit for tunable kernel functions," in *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2021, pp. 1–6.

[33] 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*, vol. 32, no. 10, pp. 7212–7225, 2022.

[34] L. Shi, C. Soell, A. Baenisch, R. Weigel, J. Seiler, and T. Ussmueller, "Concept for a cmos image sensor suited for analog image pre-processing," *arXiv preprint arXiv:1502.07449*, 2015.

[35] A. Irmanova, O. Krestinskaya, and A. P. James, "Neuromorphic adaptive edge-preserving denoising filter," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, 2017, pp. 1–6.

[36] S. Huang, T. Lu, Z. Lu, J. Rong, X. Zhao, and J. Li, "Cmos image sensor fixed pattern noise calibration scheme based on digital filtering method," *Microelectronics Journal*, vol. 124, p. 105431, 2022.

[37] W. Jendernalik, G. Blakiewicz, J. Jakusz, S. Szczepanski, and R. Piotrowski, "An analog sub-miliwatt cmos image sensor with pixel-level convolution processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 2, pp. 279–289, 2013.

[38] M. Jin, H. Noh, M. Song, and S. Y. Kim, "Design of an edge-detection cmos image sensor with built-in mask circuits," *Sensors*, vol. 20, no. 13, 2020.

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

[40] W. Jendernalik, G. Blakiewicz, J. Jakusz, S. Szczepanski, and R. Piotrowski, "An analog sub-miliwatt cmos image sensor with pixel-level convolution processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 2, pp. 279–289, 2013.

[41] G. T. Tchendjou and E. Simeu, "Self-healing imager based on detection and conciliation of defective pixels," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 251–254.

[42] A. El Gamal and H. Eltoukhy, "Cmos image sensors," *IEEE Circuits and Devices Magazine*, vol. 21, no. 3, pp. 6–20, 2005.

[43] M. Sarkar, D. S. S. Bello, C. van Hoof, and A. J. P. Theuwissen, "Biologically inspired cmos image sensor for fast motion and polarization detection," *IEEE Sensors Journal*, vol. 13, no. 3, pp. 1065–1073, 2013.

[44] T. Kuroda, *Essential principles of image sensors*. CRC press, 2017.

[45] R. J. Gove, "Cmos image sensor technology advances for mobile devices," in *High Performance Silicon Imaging*. Elsevier, 2020, pp. 185–240.

[46] M. Bigas, E. Cabruja, J. Forest, and J. Salvi, "Review of cmos image sensors," *Microelectronics Journal*, vol. 37, no. 5, pp. 433–451, 2006.

[47] T. Watabe, M. Goto, H. Ohtake, H. Maruyama, M. Abe, K. Tanioka, and N. Egami, "New signal readout method for ultrahigh-sensitivity cmos image sensor," *IEEE Transactions on Electron Devices*, vol. 50, no. 1, pp. 63–69, 2003.

[48] S. S. George, M. F. Bocko, and Z. Ignjatovic, "Current sensing-assisted active pixel sensor for high-speed cmos image sensors," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4365–4372, 2015.

[49] M. Jaklin, D. García-Lesta, V. M. Brea, and P. López, "Hdr 4t-aps pixel for event generation by frame differencing," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2021, pp. 921–924.

[50] K. Mori, N. Yasuda, K. Miyauchi, T. Isozaki, I. Takayanagi, J. Nakamura, H.-C. Chien, K. Fu, S.-G. Wuu, A. Berkovich, S. Chen, W. Gao, and C. Liu, "A 4.0 μm stacked digital pixel sensor operating in a dual quantization mode for high dynamic range," *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2957–2964, 2022.

[51] S. Leitner, H. Wang, and S. Tragoudas, "Design of scalable hardware-efficient compressive sensing image sensors," *IEEE Sensors Journal*, vol. 18, no. 2, pp. 641–651, 2018.

[52] M. Gottardi, L. Parmesan, P. Tosato, E. Demenev, E. Manuzzato, and L. Gasparini, "A 500 $\times$ 500 pixel image sensor with arbitrary number of rois per frame and image filtering for center of mass estimation," in *ESSCIRC 2023- IEEE 49th European Solid State Circuits Conference (ESSCIRC)*, 2023, pp. 97–100.

[53] M. Wayne, A. Ulku, A. Ardelean, P. Mos, C. Bruschini, and E. Charbon, "A 500 $\times$ 500 dual-gate spad imager with 100

[54] M. Gottardi, L. Parmesan, P. Tosato, E. Demenev, M. Lecca, E. Manuzzato, and L. Gasparini, "A 500 x 500 pixel image sensor with multiple regions of interest for center of mass-based event detection," *IEEE Sensors Journal*, pp. 1–1, 2024.

[55] T. Takahashi, Y. Kaji, Y. Tsukuda, S. Futami, K. Hanzawa, T. Yamauchi, P. W. Wong, F. T. Brady, P. Holden, T. Ayers, K. Mizuta, S. Ohki, K. Tatani, H. Wakabayashi, and Y. Nitta, "A stacked cmos image sensor with array-parallel adc architecture," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 1061–1070, 2018.

[56] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down," in *Power-Aware Computer Systems: Third International Workshop, PACS 2003, San Diego, CA, USA, December 1, 2003 Revised Papers 3*. Springer, 2005, pp. 26–40.

[57] T. A. Borges, L. B. Soares, and C. Meinhardt, "A fine-grained methodology for accuracy-configurable and energy-efficient gaussian filters design," in *2020 33rd Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2020, pp. 1–6.

[58] K. S. Zaman, M. B. I. Reaz, S. H. Md Ali, A. A. A. Bakar, and M. E. H. Chowdhury, "Custom hardware architectures for deep learning on portable

devices: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6068–6088, 2022.

[59] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. Nielsen, "Suitability of recent hardware accelerators (dsps, fpgas, and gpus) for computer vision and image processing algorithms," *Signal Processing: Image Communication*, vol. 68, pp. 101–119, 2018.

[60] R. Choudhury, S. R. Ahamed, and P. Guha, "Pipelined training accelerator for portable devices," *AEU - International Journal of Electronics and Communications*, vol. 177, p. 155167, 2024.

[61] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of fpga, gpu and cpu in image processing," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 126–131.

[62] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[63] L. Liu, J. Zhu, Z. Li, Y. Lu, Y. Deng, J. Han, S. Yin, and S. Wei, "A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–39, 2019.

[64] M. Kalbasi and H. Nikmehr, "A fine-grained pipelined 2-d convolver for high-performance applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 1, pp. 146–150, 2019.

[65] L. Ioannou, A. Al-Dujaili, and S. A. Fahmy, "High throughput spatial convolution filters on fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1392–1402, 2020.

[66] A. Gabiger-Rose, M. Kube, R. Weigel, and R. Rose, "An fpga-based fully synchronized design of a bilateral filter for real-time image denoising," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 8, pp. 4093–4104, 2014.

[67] D. Mukherjee and S. Mukhopadhyay, "Fast hardware architecture for 2-d separable convolution operations," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 12, pp. 2042–2046, 2018.

[68] P. Toledo, R. Rubino, F. Musolino, and P. Crovetti, "Re-thinking analog integrated circuits in digital terms: A new design concept for the iot era," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 816–822, 2021.

[69] H. Kobayashi, A. Kuwana, J. Wei, Y. Zhao, S. Katayama, T. M. Tri, M. Hirai, T. Nakatani, K. Hatayama, K. Sato, T. Ishida, T. Okamoto, and T. Ichikawa, "Analog/mixed-signal circuit testing technologies in iot era," in *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, 2020, pp. 1–4.

[70] G. Gennis, A. Kamperi, V. Alimisis, C. Dimas, and P. P. Sotiriadis, "An area-efficient, analog integrated image edge detector based on the robert's cross operator," in *2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2023, pp. 1–4.

[71] G. Gennis, V. Alimisis, C. Dimas, and P. P. Sotiriadis, "A general purpose, low power, analog integrated image edge detector, based on a current-mode gaussian function circuit," *Analog Integrated Circuits and Signal Processing*, vol. 114, no. 2, pp. 195–206, 2023.

[72] S. Lee, B. Jeong, K. Park, M. Song, and S. Y. Kim, "On-cmos image sensor processing for lane detection," *Sensors*, vol. 21, no. 11, 2021.

[73] M. Yildirim and F. Kacar, "Retina-inspired neuromorphic edge enhancing and edge detection," *AEU - International Journal of Electronics and Communications*, vol. 115, p. 153038, 2020.

[74] M. Yildirim, "Adapting laplacian based filtering in digital image processing to a retina-inspired analog image processing circuit," *Analog Integrated Circuits and Signal Processing*, vol. 100, pp. 537–545, 2019.

[75] V. Rieger, H. Schütz, S. Gambach, and A. Rothermel, "On-chip spatial filter for subretinal implants," in *2015 Nordic Circuits and Systems Conference (NORCAS): NORCHIP & International Symposium on System-on-Chip (SoC)*, 2015, pp. 1–4.

[76] J. Garcia-Lamont, "Analogue cmos prototype vision chip with prewitt edge processing," *Analog Integrated Circuits and Signal Processing*, vol. 71, no. 3, pp. 507–514, 2012.

[77] C. Soell, L. Shi, J. Roeber, M. Reichenbach, R. Weigel, and A. Hagelauer, "Low-power analog smart camera sensor for edge detection," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 4408–4412.

[78] J. Dubois, D. Ginhac, and M. Paindavoine, "A single-chip 10000 frames/s cmos sensor with in-situ 2d programmable image processing," in *2006 International Workshop on Computer Architecture for Machine Perception and Sensing*, 2006, pp. 124–129.

[79] T.-H. Hsu, Y.-K. Chen, T.-H. Wen, W.-C. Wei, Y.-R. Chen, F.-C. Chang, H. Kim, Q. Chen, B. Kim, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "A 0.5v real-time computational cmos image sensor with programmable kernel for always-on feature extraction," in *2019 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2019, pp. 33–34.

[80] Z. Chen, H. Zhu, E. Ren, Z. Liu, K. Jia, L. Luo, X. Zhang, Q. Wei, F. Qiao, X. Liu, and H. Yang, "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*, vol. 67, no. 2, pp. 389–400, 2020.

[81] G. H. Chapman, J. Leung, A. Namburete, I. Koren, and Z. Koren, "Predicting pixel defect rates based on image sensor parameters," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*. IEEE, 2011, pp. 408–416.

[82] G. H. Chapman, R. Thomas, Z. Koren, and I. Koren, "Empirical formula for rates of hot pixel defects based on pixel size, sensor area, and iso," in *Sensors, Cameras, and Systems for Industrial and Scientific Applications XIV*, vol. 8659. SPIE, 2013, pp. 119–129.

[83] G. T. Tchendjou and E. Simeu, "Detection, location and concealment of defective pixels in image sensors," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[84] Y. Chen, W. Zhu, D. Chen, and Z. Lu, "Online image sensor fault detection for autonomous vehicles," in *2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2022, pp. 120–127.

[85] Y. Cao, L. Zhang, S. S. Zalivaka, C.-H. Chang, and S. Chen, "Cmos image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 11, pp. 2629–2640, 2015.

[86] L. Mijatović, H. Dean, and M. Rožić, "Implementation of algorithm for detection and correction of defective pixels in fpga," in *2012 Proceedings of the 35th International Convention MIPRO*, 2012, pp. 1731–1735.

[87] V. Premachandran and R. Kakarala, "Measuring the effectiveness of bad pixel detection algorithms using the roc curve," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2511–2519, 2010.

[88] C.-h. Chan, "Dead pixel real-time detection method for image," Sep 2009, uS Patent 7,589,770.

[89] C.-Y. Cho, T.-M. Chen, W.-S. Wang, and C.-N. Liu, "Real-time photo sensor dead pixel detection for embedded devices," in *2011 International Conference on Digital Image Computing: Techniques and Applications*. IEEE, 2011, pp. 164–169.

[90] N. El-Yamany, "Robust defect pixel detection and correction for bayer imaging systems," *Electronic Imaging*, vol. 2017, no. 15, pp. 46–51, 2017.

[91] G. T. Tchendjou and E. Simeu, "Self-healing image sensor using defective pixel correction loop," in *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2019, pp. 1–6.

[92] G. T. Tchendjou, "Defective pixel analysis for image sensor online diagnostic and self-healing," in *2019 IEEE 37th VLSI Test Symposium (VTS)*. IEEE, 2019, pp. 1–6.

[93] L. Yongji and Y. Xiaojun, "A design of dynamic defective pixel correction for image sensor," in *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, 2020, pp. 713–716.

[94] A. Kumar, S. Sarkar, and R. Agarwal, "A novel algorithm and hardware implementation for correcting sensor non-uniformities in infrared focal plane array based staring system," *Infrared physics & technology*, vol. 50, no. 1, pp. 9–13, 2007.

[95] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 37–47.

[96] P. E. Allen and D. R. Holberg, *CMOS analog circuit design*. Elsevier, 2011.

[97] D. Ginhac, J. Dubois, B. Heyrman, and M. Paindavoine, "A high speed programmable focal-plane simd vision chip," *Analog Integrated Circuits and Signal Processing*, vol. 65, no. 3, pp. 389–398, 2010.

[98] M. Suarez, V. M. Brea, J. Fernandez-Berni, R. Carmona-Galan, D. Cabello, and A. Rodriguez-Vazquez, "Low-power cmos vision sensor for gaussian pyramid extraction," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 2, pp. 483–495, 2016.

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

[100] G. Palmisano, G. Palumbo, and S. Pennisi, "Design procedure for two-stage cmos transconductance operational amplifiers: A tutorial," *Analog Integrated Circuits and Signal Processing*, vol. 27, no. 3, pp. 179–189, 2001.

[101] B. Razavi, *Design of Analog CMOS Integrated Circuits*, ser. IRWIN ELECTRONICS & COMPUTER E. McGraw-Hill, 2017.

[102] G. Ferrari, F. Gozzini, A. Molari, and M. Sampietro, "Transimpedance amplifier for high sensitivity current measurements on nanodevices," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, pp. 1609–1616, 2009.

[103] T. Zhou, J. Liu, B. Fu, Y. Cao, Y. He, B. Jiang, and Y. Su, "A high-precision and high-linearity readout integrated circuit for infrared focal plane array applications," *Optik*, vol. 185, pp. 168–177, 2019.

[104] D. J. Comer, D. T. Comer, and A. G. Shreeve, "Linear voltage to current conversion using submicron cmos devices," *Faculty Publications*, 2004.

[105] S. Malakar, S. Gayen, S. Tewari, and A. Chattopadhyay, "Comparative analysis of performance and its stability against real-time non-ideal conditions between dg-tfet sensor and its mos equivalent for a range of biomolecule detection: a design perspective," *AEU - International Journal of Electronics and Communications*, vol. 177, p. 155242, 2024.

[106] I. Vornicu, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "Compensation of pvt variations in tof imagers with in-pixel tdc," *Sensors*, vol. 17, no. 5, 2017.

[107] A. Spivak, A. Belenky, and O. Yadid-Pecht, "Very sensitive low-noise active-reset cmos image sensor with in-pixel adc," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 10, pp. 939–943, 2016.

[108] K. Nie, X. Wang, J. Qiao, and J. Xu, "A full parallel event driven readout technique for area array spad flim image sensors," *Sensors*, vol. 16, no. 2, 2016.

[109] V. Vinayaka, S. P. Namboodiri, A. Roy, and R. J. Baker, "Segmented digital sipm," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2019, pp. 1118–1121.

[110] J. Huang, M. Guo, S. Wang, and S. Chen, "A motion sensor with on-chip pixel rendering module for optical flow gradient extraction," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[111] C. Azcona, B. Calvo, S. Celma, and N. Medrano, "Low-voltage low-power cmos rail-to-rail v-i converters," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, 2011, pp. 182–185.

[112] A. Handkiewicz, S. Szczesny, and M. Kropidtowski, "Over rail-to-rail fully differential voltage-to-current converters for nm scale cmos technology," *Analog Integrated Circuits and Signal Processing*, vol. 94, pp. 139–146, 2018.

[113] Y.-H. Sa, P.-H. Son, K.-H. Kim, H.-S. Kim, and H.-W. Cha, "A design of new voltage to current converter with high linearity and wide tuning," in *2016 International SoC Design Conference (ISOCC)*, 2016, pp. 119–120.

[114] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

[115] Y. Akhmetov, J. J. Mathew, and A. P. James, "Variable pixel g-neighbor filters," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[116] O. Krestinskaya, K. Salama, and A. James, "Analog image denoising with an adaptive memristive crossbar network," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 3453–3457.

[117] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.

[118] I. Abdou and W. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 753–763, 1979.

[119] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved canny edge detection algorithm," in *2014 IEEE International Conference on Mechatronics and Automation*, 2014, pp. 577–582.

[120] G. H. Chapman, I. Koren, Z. Koren, J. Dudas, and C. Jung, "Methods and apparatus for detecting defects in imaging arrays by image analysis," Aug 2011, uS Patent 8,009,209.

[121] M. Yang, S. Chen, X. Wu, W. Fu, and Z. Huang, "Identification and replacement of defective pixel based on matlab for ir sensor," *Frontiers of Optoelectronics in China*, vol. 4, no. 4, pp. 434–437, 2011.

[122] M. N. Sabry, H. Omran, and M. Dessouky, "Systematic design and optimization of operational transconductance amplifier using gm/id design methodology," *Microelectronics journal*, vol. 75, pp. 87–96, 2018.

[123] R. J. Baker, *CMOS: circuit design, layout, and simulation*. John Wiley & Sons, 2019.

[124] M. Soleimani, "Design and implementation of voltage - mode min/max circuits," *Journal of Engineering Science and Technology Review*, vol. 8, pp. 166–171, 2015.

[125] I. Vornicu, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "Compensation of pvt variations in tof imagers with in-pixel tdc," *Sensors*, vol. 17, no. 5, p. 1072, 2017.

[126] K. Lee, S. Park, S.-Y. Park, J. Cho, and E. Yoon, "A 272.49 pj/pixel cmos image sensor with embedded object detection and bio-inspired 2d optic flow generation for nano-air-vehicle navigation," in *2017 Symposium on VLSI Circuits*. IEEE, 2017, pp. C294–C295.

[127] K. Park, M. Song, and S. Y. Kim, "The design of a single-bit cmos image sensor for iris recognition applications," *Sensors*, vol. 18, no. 2, p. 669, 2018.

[128] M. Nazhamaiti, H. Xu, Z. Liu, Y. Chen, Q. Wei, X. Wu, and F. Qiao, "Ns-md: Near-sensor motion detection with energy harvesting image sensor for always-on visual perception," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

[129] T. Nomura, R. Zhang, and Y. Nakashima, "An energy efficient stochastic+spiking neural network," *Bulletin of Networking, Computing, Systems, and Software*, vol. 10, no. 1, pp. 30–35, 2021.