Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria Elettronica, delle Telecomunicazioni e Tecnologie dell'Informazione

Ciclo XXX

Settore Concorsuale:
Area 09 (Ingegneria Industriale e dell'Informazione) – 09/E3 Elettronica

Settore Scientifico Disciplinare:
Area 09 (Ingegneria Industriale e dell'Informazione) – ING-INF/01 Elettronica

ULTRA-LOW POWER IOT SMART VISUAL SENSING DEVICES FOR ALWAYS-ON APPLICATIONS

Presentata da: Dott. Manuele Rusci

Coordinatore Dottorato
Prof. Alessandro Vanelli Coralli

Supervisore
Prof. Luca Benini

Supervisore
Dr. Elisabetta Farella

Co- Supervisore
Dr. Davide Rossi

Esame finale anno 2018

ALMA MATER STUDIORUM
UNIVERSITÁ DI BOLOGNA

DOCTORAL THESIS

Ultra-Low Power IoT Smart Visual Sensing Devices for Always-ON Applications

Author:
Manuele RUSCI

Supervisors:
Prof. Luca BENINI
Dr. Elisabetta FARELLA
Co-Supervisor:
Dr. Davide Rossi

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Electronics, Telecommunications, and Information Technologies Engineering

DEI - Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi"

2018

UNIVERSITÁ DI BOLOGNA

Abstract

DEI - Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi"

Doctor of Philosophy

Ultra-Low Power IoT Smart Visual Sensing Devices for Always-ON Applications

by Manuele RUSCI

Due to the stringent energy requirements, bringing vision technologies into the always-on end-node devices of the Internet-of-Things results extremely challenging. A big opportunity to address the energy issues of current camera-based systems arises from the integration of mixed-signal processing circuits on the imager focal plane. Moreover, by leveraging a bio-inspired event-based data representation, sensory systems reduce sensor-to-processor bandwidth and the computational requirements for data-analytics. This work presents the design of a Smart Ultra-Low Power visual sensor architecture that couples together an ultra-low power event-based image sensor with a parallel and power-optimized digital architecture for data processing. By means of mixed-signal circuits, the imager generates a stream of address events after the extraction and binarization of spatial gradients. When targeting monitoring applications, the sensing and processing energy costs can be reduced by two orders of magnitude thanks to either the mixed-signal imaging technology, the event-based data compression and the use of event-driven computing approaches. From a system-level point of view, a context-aware power management scheme is enabled by means of a power-optimized sensor peripheral block, that requests the processor activation only when a relevant information is detected within the focal plane of the imager. When targeting a smart visual node for triggering purpose, the event-driven approach brings a 10x power reduction with respect to other presented visual systems, while leading to comparable results in terms of detection accuracy. To further enhance the recognition capabilities of the smart camera system, this work introduces the concept of event-based binarized neural networks. By coupling together the theory of binarized neural networks and focal-plane processing, a 17.8% energy reduction is demonstrated on a real-world data classification with a performance drop of 3% with respect to a baseline system featuring commercial visual sensors and a Binary Neural Network engine. Moreover, if coupling the BNN engine with the event-driven triggering detection flow, the average power consumption can be as low as the sleep power of 0.3mW in case of infrequent events, which is 8x lower than a smart camera system featuring a commercial RGB imager.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The rising Internet-of-Things (IoT) paradigm is opening new opportunities to enrich the experience of people's everyday lives [8]. In a broad sense, Internet of Things refers to a pervasive distribution of devices, which can both interact with the physical world and communicate to each other through the internet connection. At the edge of the growing IoT infrasture, an heterogeneous set of end-nodes devices, such as smart sensors and actuators, are placed to build instances of cyber-phisical systems, which include many technologies belonging to smart homes, intelligent transportation and smart cities, among the others. In other words, the IoT vision relies on a massive deployment of end-node devices to form "a worldwide network of interconnected objects uniquely addressable, based on standard communication protocols" [54]. By 2020, billions of physical devices are expected to be part of the Internet of Things ecosystem [36]. This comes together with a terrific economic impact, which is estimated around USD 2 trillion during the next five to ten years (Gartner foresee [53]).

The Internet of Things revolution will affect a wide range of application fields, spanning from agriculture to automotive, smart cities, consumer devices, retail, manufacturing, supply chain and many others [49]. Within a real-world application scenario, a network of widely distributed sensors captures data at the edge and sends information to remote central stations or servers. Any cloud service running on the server side will be capable of aggregating information from the many end-nodes and continuously update the underlying models to provide proactive feedback to the end-user [14]. Indeed, by adequately treating data, this multi-tier infrastructure can bring an unprecedented benefit to many existing services. As an example, in the context of public transportation, the offered services can be enhanced in real-time by monitoring the current occupancy and predicting the next short-term demand. The inferred information facilitate the decision process concerning the actions to take (e.g. if it is necessary to provide more vehicles) and help to prevent potential coming issues (e.g. collision between vehicles). On the same direction, the monitoring of street congestions and traffic issues can produce useful information for drivers to choose the fastest route and avoiding stacked conditions. Healthcare is another relevant field for IoT devices. Monitoring devices for vital signs and relevant biometric parameters, which may be also implanted under the skin of the patients, will enable powerful caring services for early diagnosis, treatment assistance or diseases prevention, or even remote supervision services without any impact on the individual independence.

At the present, Internet of Things is growing and getting a reality. The interconnected devices, namely the **IoT End-Nodes, stand as the basic building blocks of**

the emerging infrastructure [60]. Among the others, multiple kinds of sensors, actuators and readers can be located at the edge of the network for sensing purpose and data collection. Ideally, any IoT end-node should be able to perform perpetually its assigned task when abandoned in the environment, without demanding any periodic maintenance or replacement. Indeed, a suitable IoT node is expected to be autonomous either in terms of functionality, energy, network connectivity and possibly low-cost [3]. In the perspective of implementing a pervasive network of untethered and autonomous devices, an IoT end-node requires being powered by batteries or energy harvesters. Depending on the application environment, the battery recharge or replacement may be prevented for technical reasons or, however, should occur with a low frequency to keep the maintenance costs contained (e.g. once every one or more years). In this context, it is therefore essential to target a power-optimized design at the device level to guarantee a long duration of the battery according to the provided specification. This also includes the implementation of aggressive power savings techniques to avoid any kind of energy waste within the system. Given that, it is clear that **energy consumption is a key issue when designing an IoT end-node** [55].

To understand the design space and which are the potential knobs for regulating the power consumption, it is necessary to delve into deep of system architectures of IoT devices and their typical operating modes. Figure 1.1 illustrates the block diagram of the main sub-systems of an IoT node: sensing, processing, communication and power supply modules [23]. The *sensing unit* includes a mixed signal circuit for transforming the transduced analog signal into a digital signal that can be handled by a digital engine. To this scope, the sensor sub-system contains a signal conditioning block and an Analog-to-Digital Converter (ADC). The *processing unit* is the core of the system and coordinates the operation of all the other blocks. A digital engine is capable of running data analitycs on data coming from the sensors and dispatches externally the extracted high-level information through the *communication unit*. In contrast to simply streaming raw sensed data for off-node analysis, the node itself performs *data processing locally*, henge bringing *intelligence near to the sensor*. To highlight the importance and the advantages of local data processing, two fundamental points must be considered.

1. The Internet-of-Things network infrastructure has to be capable of handling the data traffic in a reliable and secure way. Due to the exponential increase of flowing data, the network services can deteriorate or be affected by networking issues in the upcoming future. To reduce the dimensionality of exchanged data, data mining techniques can be applied to extract hidden high-level information from the sensor data [19]. Therefore, local data filtering provides an opportunity for both reducing the wireless transmission bandwidth along with avoiding network congestion issues. *By moving data analytics at the edge, a smarter object ecosystem can aggregate and filter data locally and drastically reduce the amount of information circulating within the network.*

2. From an energy perspective, streaming raw sensed data to a base-station does not come for free. The transmission energy cost is generally high and increases linearly with the bandwidth. This represents a bottleneck for sensory systems characterized by high datarates (e.g. cameras) because the communication sub-system features a higher energy consumption than the other system components when continuously streaming raw sensor data. Indeed, despite the advances achieved during the last 5-10 years, the current absorbed by the radio module during the transmission phase can reach peaks above 10mA [46]

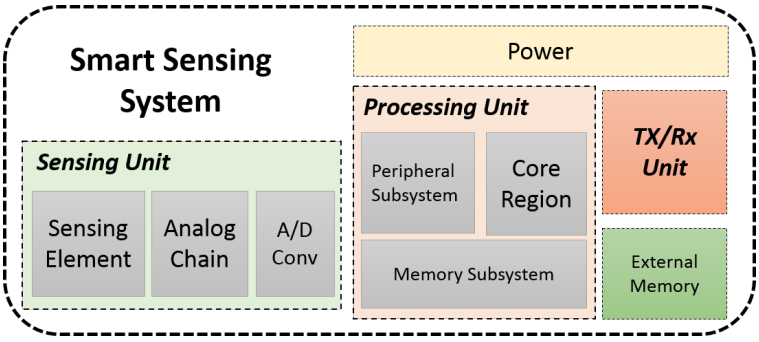


FIGURE 1.1: IoT end-node architecture.

while the digital processing logic can consume less than few mWs [6]. Within this scenario, *reducing the sensor data dimensionality by means of local "in-node" processing enables a duty-cycled usage of the communication subsystem and leads to a notable reduction of the average system power consumption*.

The above reasons justify the **necessity of moving data analytics into the sensor nodes**. By pushing intelligence at the edge, the IoT ecosystem can be enriched by "smart" things, which generate more structured and meaningful information than simply dispatching raw data. To this aim, platforms and algorithms for data analytics become central within the design process. Power-optimized devices, which feature a high computational power within a limited energy budget, enable even complex algorithms to run on resource-constrained end-nodes in real-time [28]. By applying compression algorithms, such as for pattern recognition and classification, on the sensed data, an IoT node can reduce the communication payload even to a single-bit, e.g. to trigger the detection of an object crossing a virtual gate or a specific human gesture recognition.

Typically, an IoT Smart node performs a periodic task, composed of data acquisition, processing and transmission [4]. To save energy, the system is kept in a fully-active state only for a short time T_P with an energy cost E_{act}. During the rest of the period, the node can be put in (deep) sleep mode, where only a minimal part of the system, namely the *always-on region*, is still active to trigger the wakeup of the rest of the system. In this state, the system consumes a power P_{alwON}, which is several orders of magnitude lower than the active power. Hence, the average power consumption over the period can be expressed as $P_{avg} = P_{alwON} + E_{act}/T_P$. This power consumption has to be kept as low as possible to gain an extended battery life $T_{node} = E_{battery}/P_{avg}$. Given that, a power-optimized design process of an IoT node aims at (a) reducing the average power P_{alwON} of the always-on circuitry constantly kept in a running state, (b) minimizing the energy E_{act} due to the active task but also (c) reducing the latency T_P related to the system activation rate. Figure 1.2 illustrates how varies the battery-life depending on the average power consumption. The plot shows that to reach the target of one or several years of battery lifetime, *an IoT end-node should draw a power consumption in the μW range*. It is also straightforward that a larger battery can lead to an increased lifetime, but this comes together with a larger node dimension and costs.

To meet the energy requirement, it should be clearly pointed out that saving energy by constraining node functionalities can come at the cost of performance and quality degradation [91]. As a simple example, a system that stays in idle or sleep state for the majority of the time and rarely activates for sampling and processing

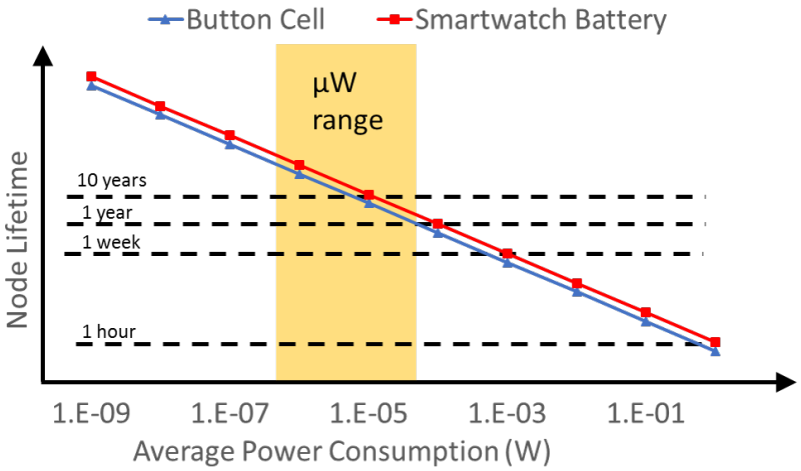


FIGURE 1.2: Battery Lifetime of an IoT end-node when powered by small batteries [3].

consumes less than a system featuring higher sampling rates but suffers from possibly miss-detections and long latencies. Therefore, **sensing quality must be taken into account when discussing energy optimization**.

1.1.1 Design Space and Challenges

To address the energy-quality trade-off concerning the design of smart visual IoT end-node, the following design aspects need to be analyzed.

- **Hardware-Software Co-Design**. At the hardware level, the components choice needs to be directed towards low-power but energy-efficient devices. From a functional viewpoint, both processing and sensing units are expected to provide the needed computational resources to accomplish the targeted task. But, on the other side, the power and energy costs increase with the complexity and flexibility of the selected components. To address it, a synergetic and power-optimized Hardware-Software design process needs to trade-off between functionalities and energy requirements of the targeted platform, also considering the partitioning between the hardware and software stacks.

- **Data Management and Analysis**. From an energetic perspective, the ability of treating data in an efficient way becomes fundamental. A tremendous energy savings can be achieved by making use of architectural solutions that support low-cost operations (e.g. a low energy costs for every memory access) or by designing lightweight data analytics models, which results into low-latency tasks demanding a low energy cost. Hence, a crucial way for addressing the energy issues at the system-level relies on refining or even re-designing models for data analytics based on the application target, together with providing an optimized implementation, which fits the memory and computational constrains of resource-constrained devices. In addition to this, but not less important, the sensing quality must not be degraded alongside the optimization process.

- **System Integration**. The integration of heterogeneous components within a single device has to be driven by energy metrics. In this context, an efficient

power management scheme has to be defined at the top level to guarantee a high sensing quality and, at the same time, to minimize the energy waste of the single components. Ideally, every part should be activated only when necessary and run with the minimal amount of energy to accomplish its assigned task.

1.2 Smart Visual Sensing Devices

Among the existing sensing technologies for the IoT ecosystem, vision is attractive because of the richness of sensed data. A distributed placement of camera sensors enables a wide range of applications in the context of, among the others, smart cities, security and environmental monitoring [107]. Recently, wearable and mobile cameras are also becoming widespread for gaming, augmented reality and consumer applications [5, 9]. Despite the great interest, the majority of the existing smart camera systems consist of power-hungry components, which prevent the use of batteries as power sources. Indeed, smart visual end-nodes typically include high-resolution imagers (i.e. several Mpixels) and high-end processing units (e.g. ARM Cortex-A processors but also FPGAs and GPUs) to handle the large amount of multidimensional visual data and the complexity of computer vision algorithms [13, 88, 33, 89]. These design choices traduce into a high system power cost, spanning from several hundreds of mW to several Watts [102]. As a consequence, when battery powered, a smart camera node's lifetime is restricted to few days or even much less, which does not match the requirements of some applications (e.g. the experience of the Google Glass [71]).

Instead, to meet the energy requirement discussed above, resource-constrained devices have to be considered during the design process of the IoT end-nodes. A viable solution includes the usage of imagers with small resolution (e.g. [52]) and low-end processing units (e.g. ARM Cortex-M Family [7]). These latter ones are characterized by a limited memory footprint (generally below 512kB) and a limited computational power as provided by a single-core architecture with a system clock running up to tens of MHz. However, these hardware limitations are justified by the ultra-low power cost, which can be as low as 10 μA/MHz [6]. When deploying computer vision applications on resource-constrained platforms, the implementation process has to be carefully addressed to optimally exploit the underlying resources. Highly-promising but computationally-demanding visual data processing implementations may not fit into the limited memory footprint or can present a latency that is too high to run in real-time. **Therefore, the design process needs to address the implementation of data analysis tasks by keeping into account the resource constraints as part of the energy-accuracy trade-off**.

1.2.1 Aim of the Research Work

At the present, the plethora of state-of-the-art smart visual sensing devices for the IoT ecosystem, which are extensively revised in Chapter 2, is missing of solutions bridging the quality of leading edge computer vision models with the energy requirements of autonomous battery-powered devices.

In this context, this thesis work aims at developing novel system-level design strategies and techniques for filling the existing gap. More in details, *this research work targets advanced visual sensing solutions featuring both (a) an average power consumption in the envelope of few mWs or sub-mW, such as a camera system can last for years*

when battery powered, and (b) smart embedded capabilities by leveraging state-of-the-art computer vision models for data processing. To reach this aim, novel techniques, even in contrast with traditional visual sensing device approaches, are needed for addressing the major design challenges described in Section 1.1.1.

1.3 Thesis Contribution

This work introduces **a Smart Visual Sensing End-Node design which leverages the Event-Based Paradigm to extremely raise the system energy-efficiency**. In contrast with traditional smart camera designs, the proposed system includes a novel ultra-low-power imager that generates a binary spatial-contrast information and dispatches only the (x,y) addresses of the asserted pixels after frame difference, i.e. the generated *events*. The imager internal processing is enabled by the hardware integration of specialized mixed-signal circuits on the focal plane. This design choice allows reducing the power cost of the imaging task and the sensor-to-processor bandwidth. Indeed, only a relevant information is transferred to the processing unit's memory in correspondence of moving objects in the camera field of view. The imager is coupled with a quad-core IoT processor, which is implemented in 28nm technology and optimized for low-power consumption.

As a first major contribution, this work describes the HW-SW co-design and the system architecture of an Event-Based visual system. Event-Based sensing and processing techniques are evaluated for moving object detection applications. An optimized implementation on a resource-constrained device is described along with the energy evaluation. The results show an energy reduction of more than two orders of magnitude with respect to low-power off-the-shelf architectures that make use of traditional computer vision approaches for object detection.

Moving towards a system-level perspective, the interest falls into the power management scheme to be exploited when dealing with event-based architectures. To reduce energy wastes, the transition from sleep to the active state of the digital processing unit is driven by the event rate of the sensor. Indeed, it is extremely inefficient to wake-up and transfer data to the processing unit when the amount of generated events is too low, i.e. a very limited motion is detected in the field of view. **The second major contribution of the thesis regards the specification, design and prototype of a camera peripheral IP, which is able both to handle the data transfer from sensor to the processor and to drive the power management scheme of the whole system.** This strategy is referred to as *event-driven power management*, because, differently from a traditional frame-driven model of camera-based systems, the wake-up rate of the processing unit depends on the context information (i.e. *context-aware*) concerning the detected motion, which is triggered by the event generation process. To demonstrate the benefits of the proposed approach, the event-based camera system is evaluated as a smart visual trigger on always-on monitoring applications. From an energy viewpoint, an energy reduction of more than 10x is measured against other proposed architectures.

To enhance the recognition capabilities but still keeping the energy consumption low, **this work also introduces the concept of Event-Based Binarized Neural Network, which is the third major contribution of the thesis. The idea consists of coupling the benefits brought by the event-based paradigm with the recognition capabilities of deep learning techniques optimized for low-power and low-end systems, the Binarized Neural Networks (BNNs).** In the context of always-on vision

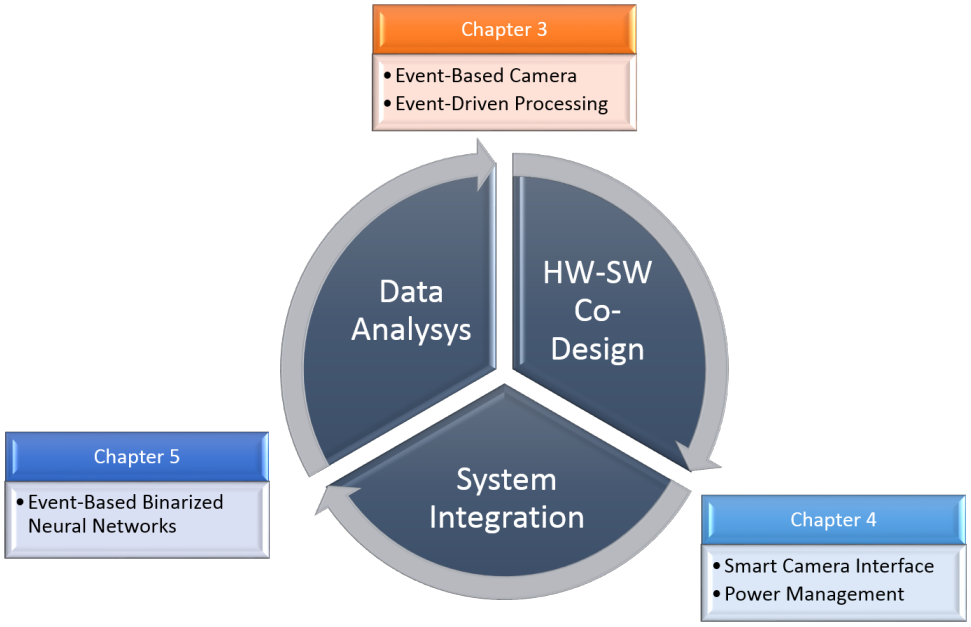


FIGURE 1.3: Thesis Content Flowchart.

systems, an experimental study is conducted for demonstrating a classification accuracy above 80% when training a BNN with binary input, on a 3-classes real-world data. Also, a BNN optimized implementation on a 4-core processor is presented to enable the proposed approach running in real-time on a resource-constrained device. The experimental result shows that this technique allows saving 17.8% energy with respect to a visual system featuring an RGB traditional imager, at a cost of 3% of classification accuracy drop. When running the classification BNN upon the trigger generated by the event-driven processing, the energy saving raises up to 8x with respect to the baseline system.

1.4 Outline of the work

The work is organized as follows.

- Chapter 2 reviews the technologies for low-power smart visual systems. A discussion regarding the state-of-the-art approaches for sensing and processing is provided here, along with a description of existing camera-based architectures. Moreover, this chapter reports previous work on Event-Based sensory systems.
- Chapter 3 focuses on Event-Based sensing and processing approaches and highlights the energy benefits when they are used for ultra-low power visual sensing. The HW-SW co-design strategies for enabling this paradigm are described in this Chapter, alongside the algorithm implementation of lightweight event-based processing models.
- Chapter 4 discusses power management issues from a system-level viewpoint. To enable an event-driven power management scheme, a description of an HW camera interface is provided here. Additionally, the chapter describes the power-optimized camera system integration along with a complete evaluation concerning the energy-accuracy trade-off when using the system as a smart visual trigger.

- Chapter 5 presents the concept of Event-Based Binarized Neural Network, showing the implementation on a resource-constrained architecture and the advantages of this kind of approach with respect to a traditional baseline camera-based systems.
- Chapter 6 summarizes the findings of the thesis work and provides some perspective directions in the field of energy-efficient smart visual sensing devices.

Figure 1.3 graphically illustrates a flowchart of the themes discussed within the thesis in response to the issues highlighted in Section 1.2.

Chapter 2

Technologies for low-power smart visual sensing

2.1 Overview

Visual IoT End-Nodes stand as the building blocks of large camera-based networked systems. These devices enable a wide range of applications, including surveillance, environmental monitoring, detection of dangerous conditions and advanced driver-assistance systems. With respect to other mono-dimensional signal sources, cameras generate a richer multi-dimensional data, which is unique for inferring high-level information from the surrounding environment. In the perspective of building a low-level sensing tier of visual nodes, which are placed in the environment and "forgot", every device should be powered by energy-harvesters or small batteries to ideally last for years while performing its assigned tasks and without demanding any maintenance intervention. Within this scenario, the power consumption of a camera-based system needs to be in the μW range to gain a lifetime of years on a small battery [3]. Therefore, **the design process of platforms for continuous and always-on visual sensing needs to be strongly optimized to meet the energy requirement dictated by the application and the network architecture**.

Placing a vast amount of camera-based sensors in the environment leads to a proliferation of data flowing through the network. For this reason, building a network architecture that supports the streaming of raw-data from the single sensors to a central station results to be weakly scalable and possibly causing network congestions. In addition, other issues arise when considering privacy aspects of exchanging visual raw data and the costly computing resources on the remote servers. Moreover, within the end-node device, the transmission subsystem is commonly the energy bottleneck of the system. This is difficult to scale due to physical limitation and causes a battery discharging in a very short time in case of continuous wireless streaming of raw data [48].

An efficient solution to address the aforementioned issues consists of bringing *intelligence* close to the sensor. According to the **Near-Sensor Processing** paradigm, the sensed data are locally analyzed *in-the-node* by means of the embedded computing capabilities. Locally filtering data reduces the data dimensionality and decreases the node's transmission bandwidth with respect to stream out all the raw data. At the same time, this leads to a notable reduction of the transmission energy cost, because the communication subsystem is activated only for a short time to transmit the high-level information extracted from the visual signal. In the extreme scenario, the transmission payload can be reduced to even a single bit which is asynchronously triggered when a specific event of interest occurs.

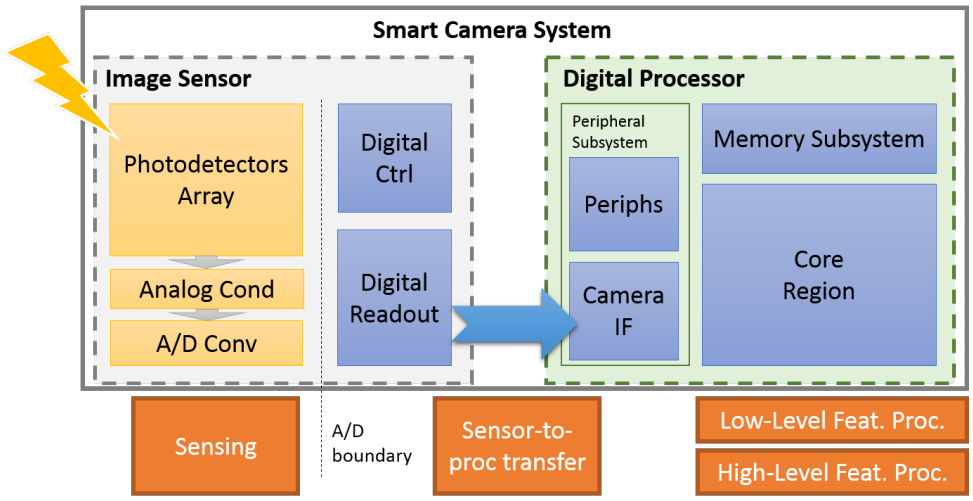


FIGURE 2.1: HW Architecture of a traditional Smart Camera system.

When featuring processing capabilities, a visual node is characterized by a more complex architectural design than a streaming camera system. Typically, such a camera node is labeled as *Smart*, as indicating a system that is not only able to capture data but also can process it thanks to a dedicated processing sub-system for data analytics. Due to the energy constraints of battery-powered devices, a data analytics workload, which typically consists of computationally-demanding *Computer Vision* algorithms, must run efficiently on the data processing engine. But, on the other hand, to reduce the energy consumption, an optimized system design process includes low-end sensing and processing units, which are characterized by a limited computational power and memory resource. Because of this trade-off, **the design process of Ultra-Low-Power Smart Camera systems becomes extremely challenging**. More specifically:

- A power-optimized Smart Camera architecture must efficiently sustain the processing workload in addition to carry out other system functionalities in an efficient way, such as visual acquisition, sensor-to-processor data transfer and power management tasks.
- The implementation of the visual processing pipeline must be optimized to run efficiently on a resource-constrained device. This includes making use of lightweight or approximated models to reduce latency and energy consumption without leading to sensing quality degradation.

From a system-level viewpoint, a *Smart Camera* system combines in a compact embedded platform the image sensing and the local visual processing engine [10]. Hence, a smart visual system is mainly composed by a sensing unit, i.e. the image sensor, and a processing sub-system, which runs data analysis on the sensor data. Optionally, an external memory can be included at the board-level to deal with the memory requirement of sensing and processing tasks. A block diagram of a typical hardware architecture is depicted in Figure 2.1.

Within the system architecture, the image sensor digitizes the visual signal before dispatching data to the processing unit for data analytics. When dealing with continuous monitoring, the image source is kept always-on, therefore the power cost for producing and digitally converting the data needs to be extremely contained.

Moreover, the sensor-to-processor data transfer, whose bandwidth linearly increases with the amount of produced data, determines an additional energy consumption, impacting the total budget. Any compression scheme aiming at reducing the bandwidth can lead to potential energy savings. A detailed review of existing low-power visual sensing technologies is provided in Section 2.2, along with proposed techniques for reducing the energy and power consumption.

The digital processor is the core of the systems and is responsible for running computer vision algorithms on the visual data. Typically, some low-level features are firstly extracted from the visual signal and then aggregated into a high-level information through classification or regression models. To this aim, a processing platform features a heterogeneous set of digital engines, which may include general-purpose processors coupled with specialized HW accelerators, a memory region and a peripheral subsystem, for interfacing the processor with external sensors. The *computational power* of the processing unit is quantified as Operations per Second (OP/S), which scales almost linearly with the clock frequency and the parallelism degree. The *energy efficiency* is another key metric that is computed as Energy per Operation (J/OP), also corresponding to the power consumption spent to provide a given computational power. To perform data analytics, a processing unit needs to provide a sufficient computational power to run the computer vision workload under the energy-efficiency constraints. Many kinds of processing units do exist and can be featured by smart camera systems. Among the others, hardware specialized processing circuits feature an energy-efficiency much higher than software-programmable digital processors but lack flexibility, that can be essential to run different types of applications on a given platform. Section 2.3 provides an overview of system architecture of smart camera systems optimized for ultra-low-power consumption.

From a system-level viewpoint, an optimized power management scheme is also essential to limit the energy wastes of different blocks. To gain efficiency, any component should be put in a low-power sleep state when idling. Typically, the processing unit handles the system timing and controls the power management policy by disabling the different components if not in a working state. Moreover, some novel power-efficient approaches can drawn inspiration from neuromorphic sensing and processing techniques, which are the subject of the review in Section 2.4.

2.2 Energy-Efficient Image Sensors

Imaging technology transforms the light signal into a digital format. The basic architecture of an image sensor is illustrated in Figure 2.1. The sensing transducer is composed by a a matrix of photodetectors, which produces a 2D analog visual signal feeding an analog chain serving for amplification and digital conversion. Generally, image sensors feature a single or a column-parallel analog chain, therefore the photodetectors matrix needs to be sequentially scanned to digitize large 2D arrays. Previous studies have confirmed that photodiodes arrays can feature a power cost as low as few μW [50], while the analog chain requires orders of magnitude higher power [72]. Outside of the analog domain, a digital controller handles the timing of the pixel exposure and of the signal conversion. The digital domain can also include a processing pipeline for filtering or adjusting the sampled values (e.g denoising) and a readout module that manages IO operations.

Among the leading commercial power-optimized image sensors, *AMS International AG/Awaiba NanEye2D* [86] and *OmniVision* samples [87] feature a compact

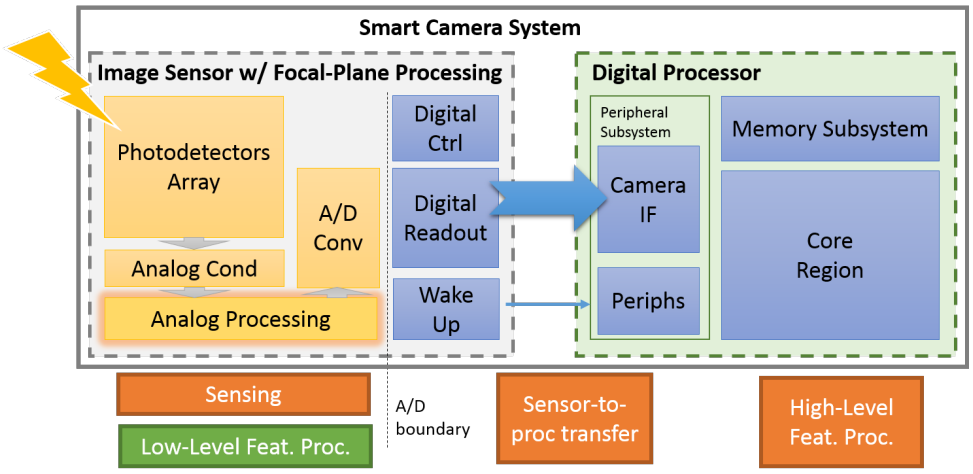


FIGURE 2.2: HW Architecture of a Smart Camera system featuring an image sensor with Focal-Plane Processing capabilities.

form factor and generates (sub-)VGA pixel images at a power cost of few tens of mWs. More recently, *Himax* [52] presented an image sensor for always-on applications with 160x120 resolution at the lowest power consumption of 1.1mW at 30fps. Along with them, it is worth to mention the *Stonyman* camera from *CentEye* [18], which provide as output the analog visual signal and can be employed for embedded recognition systems [27].

Going into deep of typical image sensor architectures, the acquisition pipeline is sized such as to sample the high-dimensional data with high-fidelity and resolution. However, such a signal is known to be largely redundant as, in a video stream, it presents either high spatial and temporal correlations. Moreover, when dealing with detection or classification tasks, only a subset of a few but significant features result meaningful to solve the decisional problem [101]. Due to these motivations, a viable way to cut the majority of the sensing energy costs relies on digitally converting only the set of visual features used to feed computer vision models. To implement this strategy, **a feature-extraction layer needs to be pushed directly on the focal-plane of the image sensors**, therefore opening the way for new architectural solutions [94], usually referred as **Focal-Plane Processing**. In fact, a mixed-signal processing circuit, placed early in the analog chain, enables a first data compression and extraction of low-level features on the sensor die, hence addressing the data dimensionality reduction within the analog domain. From an energy viewpoint, the costly analog-to-digital sub-system can be activated for a reduced time than for converting the entire raw 2D signal, due to the lower amount of data to be digitized, and can optionally support a reduced bit-precision [85]. Given that, an image sensor with early-processing capabilities can feature a simplified analog-to-digital chain design to reduce the sensor power consumption and bring the imager power cost close or within the μW range.

Different strategies of Focal-Plane Processing have been already presented. Figure 2.2 illustrates the block diagram of an imager with focal-plane processing capabilities within a smart camera architecture. The figure highlights the presence of an analog (mixed-signal) processing blocks inside the sensing units, aiming at early-extracting low-level features from the visual signal. The extracted features will be then digitally sampled through the ADC converter and transferred to the processing system by the readout interface. It is important to note that, in the perspective

TABLE 2.1: Focal Plane Processing Imagers

Ref	[58]	[24]	[39]	[100]	[47]	[66]	[20]	[73]
Function	Motion Detection	Motion Detection + HOG Extraction	Spatial filtering (4-conn)	Gaussia Pyramids	Spatial Temporal Filtering	Optical Convolution	Analog Multiply (front-end only)	Stacked Convolution
Output format	1b(mot) 9b(image)	1b(mot) 8b(HOG)	8b	8b	1b (events)	8b	6b	10b
Array	128x128	128x128	174x144	176x120	128x64	96x64	no array	227x227
Tech	0.13μm	0.18μm	0.35μm	0.18μm	0.18μm	0.18μm	40nm	0.18μm
Power Cost	467nW (motion) 16μW (image)	220nW (motion) 2.4μW (HOG)	2.9mW (4x4 filters)	70mW	100μW	1.8mW	228μW	42mW (sim)

of reducing the total system-energy, this approach results beneficial with respect to a traditional smart camera system of Figure 2.1, because of (a) the *reduced sensor-to-processor bandwidth* and (b) the *lower demand for digital computation*. Indeed, the focal-plane processing approach implies moving the low-level feature-extraction operations from the digital processor to the analog side. The early-extracted features can also feed a wake-up block, which is included into the imager, to produce a wake-up trigger signal in case of detection of interesting events within the camera field of view.

Multiple flavors of Focal Plane Processing circuits have been proposed for low-power sensing [47, 24, 58, 37, 100, 66, 73, 20]. Table 2.1 reports the more relevant works. An ultra-low-power imager for always-on application is proposed by Kim et al. [58]. The design features a 128x128 array with in-pixel motion detection realized through analog frame difference. The fabricated sample consumes 467nW at 5fps. Viceversa, in imaging mode, the power cost increases up to 16μW at 6.4fps. In [24], authors proposed the integration of HOG feature extractor circuits within the pixel array. The HOG mixed-signal computation is triggered by motion detection. To this aim, a motion bitmap is generated at a very low-power cost (<220nW) by thresholding the output of the frame difference between the current signal and the last binary frame, which is stored through in-pixel capacitors. Once the motion is detected, the sensor dispatch externally an 8-bit HOG feature signals, while consuming 3.4μW /frame. *FLIP-Q* [37] is a 176x144 imager featuring pixel-wise analog processing elements. These latter enable spatial filtering and subsampling over rectangular-shaped patches. The processing is performed on 4-connected pixel groups and serves for generating a configurable multi-resolution scene representation. Every connection can be individually enabled or disabled. The output values from the analog blocks are then converted by a 8-bit SAR ADC. The measured power consumption can be as low as 2.9mW when applying spatial filtering on 4x4 blocks. More recently, the front-end [100] permits to compute the 8-bit Gaussian pyramids on the sensor die, by integrating a matrix of 88x60 analog processing elements, each one operating on a 2x2 pixel patch (the pixel array is 176x120). The Gaussian filtering, which is employed within object detection inference pipelines, is realized using a diffusive and switched-capacitor grid. The power cost including the analog-to-digital conversion is measured as 70mW, which outperforms architectures composed by traditional imager and an external mid-high end processor that runs the same task. Another relevant example of focal-plane processing is demonstrated by the imager presented by Gottardi et al [47]. The proposed design embeds (a)

mixed-signal circuits to extract binary spatial gradients and (b) a frame-difference scheme to detect the moving edges on the camera plane. Differently the other solutions, this imager features an event-based representation to compress and dispatch the data out from the sensor. This compression scheme consists of sending only the (x,y) addresses of the asserted binary pixels of the difference map. This is strongly related to the neuromorphic principles illustrated in Section 2.4 and it will be later better analyzed. Due to this unique combination, the proposed imager functionalities are further explored within this thesis work. A more detailed description is also provided in Section 3.2.1.

Given the raised interest for brain-inspired computer vision techniques, such as deep learning approaches, recent solutions have tried to push specialized analog circuits on the focal plane to enable an early and energy-efficient computation of common functions, such as image convolutions [20, 66, 73]. The work presented in [20] makes use of angle-sensitive pixels, which integrate diffraction gratings. Based on the different orientations of the pixel-level filters, multiple convolutional feature maps are optically computed as the outcome of the first convolutional output layer. The total power cost results equal to 1.8mW. Also, to enable early-convolution, a sensing front-end proposed in [66] can perform analog multiplication. Authors introduce a MAC unit composed of only passive switches and capacitors to realize a switched-capacitor matrix multiplier, which achieves an energy efficiency of 8.7 TOp/s/W, when running convolution operations, and a power cost of $228\mu W$. Besides them, *RedEye* [73] embeds column-wise processing pipelines in the analog domain to perform 3D convolutions before of the digital conversion. The analog processing circuits can perform convolution and pooling operations. A digitally-clocked controller is responsible for handling the cyclic reuse of the computation block to serve multiple stacked convolutional layers. Weights are stored in an internal memory with a maximum 8-bit resolution and converted in analog at runtime. The chip is implemented in 0.18 μm technology and needs 1.4 mJ to process the initial 5 layers of GoogLeNet at 30fps, leading to an energy efficiency of less than 2 TOp/s/W.

2.3 Smart Camera Nodes and Local Data Processing

When targeting computer vision applications, a smart camera system shall feature (a) *high computational power*, for running data processing tasks, (b) *programmability*, to favor system flexibility and reduce the development time of a given product, and (c) *ultra-low-power consumption*, to match the energy requirement of battery-powered devices. To enable local data processing, multiple kinds of digital processing units can be included within a smart visual system. Among the software-programmable platforms, *Graphic Processing Units* (GPUs) provide a huge computational power by exploiting a highly parallel architecture. However, this comes at the cost of a high power consumption, within a range of tens of Watts, which is too high in the perspective of being powered by batteries. A largely diffused solution for smart camera systems leverages mid-to-high end *Central Processing Unit* (CPU) architectures, e.g. 32-bit and 64-bit RISC *ARM Cortex-A* processors [7]. The software-programmability and the high-clock frequencies (hundreds of MHz) featured by these platforms enable the implementation of many computer vision algorithms running with low-latencies. But still, the power consumption can raise up to hundreds of mW [21]. At the other side of the spectrum, *MicroController Units* (MCUs) have become largely widespread as low-power embedded processors (e.g. *STM32 ARM Cortex-M* Family

TABLE 2.2: Smart Camera Systems

Ref	Eye of Things [33]	CMUcam5 [25]	CITRIC [21]	Cyclops [92]	MeshEye [100]	Wi-FLIP [40]	[16]	[43]
Imager	NanEye2D [86] Himax HM01B0 [52]	OmniVi. OV9715 (1280x800) [87]	OmniVi, OV9655 1280x1024, 640x480 [87]	Agilent ADNS-3060 (30x30), Agilent ADCM-2700 (640x480)	Agilent ADCM-1700 (352x288)	FLIP-Q [37]	SCAMP [35] (128x128 w/ focal plane processing)	[47] (64x128 w/ focal plane processing)
Processor	Movidius Myriad2 MA2450	NXP LPC4330	Intel XScale PXA2700	ATMEL ATmega128L (8-bit RISC)	ATMEL AT91 (32-bit)	Intel XScale PXA2700	IGLOO FPGA + NXP LPC1769	IGLOO FPGA
Features	Parallel SIMD + HW acc	Dedicated engine for the extraction of high-level information	Image down-sampling and cropping at the hardware level	CPLD frame-grabber halted by the MCU, as well as the external SRAM	High-resolution camera acquisition is triggered once an object is detected and stereo matched	Focal Plane Proc.	Focal Plane Proc. - FPGA wakes-up by the MCU after image acquisition	Focal Plane Proc. - Imager sleep mode - Event Driven Camera IF with dual clock domain and clock-gating
Data Processing	CV Libraries for detection and recognition	Color-based Object Detection, Connected Components	Object Detection and Tracking	Optimized Kernels for motion detection and bckg subtraction	Object Detection, Stereo matching, object acquisition	Triggering of alert conditions	Object detection and counting - Smart trigger	Single people counter
Power Cost	>1W	700mW (typical 140mA @ 5V)	751mW	33mW	175.9mW [102]	>100mW	5.5mW	2.5-4.2 mW

[99]). These systems include a flexible peripheral subsystem and on-chip memories, either volatile or not volatile, along with one or more CPUs. Off-the-shelf MCU devices present a power consumption as low as 10 μW /MHz [6]. Despite the low cost, MCUs feature limited computational power (the clock frequency is typically up to few tens of MHz) and memory resources (typically up to 128kB), which may not be sufficient to sustain the requirement of some data processing algorithms. Therefore, the design and implementation of computer vision algorithms on these devices results challenging and need to be highly optimized.

Recent surveys list camera nodes both from academia and industry [102, 2, 38]. Table 2.2 reports the principal features of the most relevant power-optimized smart camera systems. Among the commercial flexible nodes, *OpenMV* [88], *JeVOIS* [56] and *RapsberryPI* [89] show low-cost and high flexibility thanks to a featured high-level programming models, supporting, among the others, *Python* or *OpenCV* libraries. The *Blackfin Low Power Imaging Platform (BLIP)* [13] includes a powerful

Blackfin embedded processor running up to 400 MHz and a VGA imager by *Omnivision* [87]. Another design that includes off-the-shelf components is *CMUcam5* [25], which embeds a *NXP LPC4330* processing unit. Despite the high flexibility given by the high-end processing units and the wide range of algorithms that can be implemented, these designs show a power consumption of several hundreds of mW on typical application workload.

Additionally, among the systems including high-end processors, the smart camera *CITRIC* [21] has been employed for object detection and tracking with a power budget around 700mW. This platform couples an *Intel XScale PXA2700* 32-bit processor with a 1280x1024 CMOS image sensor *OV9655* by *Omnivision*. The processor runs at a maximum speed of 624 MHz and includes a specialized camera interface. To reduce the energy when running computer vision tasks, some HW-SW optimizations have been tested on the platform [17]. By means of HW down-sampling and cropping, and performing SW detection and tracking in cropped regions, a 41% decrease in energy consumption and a 107% increase in battery-life can be obtained. A more recent platform but still presenting a high power consumption is *Eye of Things* [33]. This system features high computational power and flexibility thanks to the processor *Myriad2 MA2450* by *Movidius*. This processing unit is a heterogeneous multicore processor that includes twelve 128-bit Very Long Instruction Word *SHAVE* processors, two 32-bit RISC processors and a hardware acceleration pipeline coupled with a shared memory subsystem and peripherals. Thanks to the *SHAVE* coprocessors and the video HW accelerators, the platform can sustain a workload up to 1000 GFLOPs (16-bit fp type). The chip is designed to work at 0.9V and 600MHz at a power cost of 600mW, which includes peripherals and the external 512MB DDR3 RAM. The *Eye of Things* node supports either *NanEye2D* [86] and *Himax HM01B0* [52] sensors. To eases the development of computer vision algorithms, the supported programming framework includes low-level computing kernels which efficiently exploit the underlying parallel hardware. When running a Deep Learning inference task, authors showed a 1.1W power consumption at board-level.

Moving to the other side of the spectrum, *Cyclops* [92] features an *ATMEL ATmega128L* 8-bit processor together with a CIF-resolution (352x288) camera. An additional CPLD is placed between the processing and the sensing units and acts as a frame-grabber when enabled by the MCU. The system features a basic power management strategy by staying in sleep mode when waiting for a frame acquisition. A power consumption of 33mW is reported when the node is employed for triggering wake-up signals to upper layer network systems based on motion detection [63]. Despite the low power, the processor works with a clock frequency of 7.3MHz, leading to a limited computational power (computation takes 240msec for basic background subtraction and motion detection algorithms on 128x128 images).

A design approach to reduce system energy consumption employs a low power early-detection pipeline to trigger the activation of power-hungry cameras and complex visual analysis [51, 84, 77]. For instance, the visual system presented by Magno et al. [77] contains a Passive Infra-Red (PIR) sensor that is used as always-on sensing source to trigger a camera record. The PIR detects heat variations due to human presences and movements. Based on the absence/presence of people in the scene, the PIR can activate the camera recording and the local processing. The software-based data analysis is optimized to run on a constrained processor, lacking Floating Point Unit (FPU) coprocessors, and consists of background subtraction, clustering and labeling to detect abandoned/removed objects. As a relevant outcome of this

work, the battery lifetime depends on the event rate, as triggered by the PIR sensor. Indeed, the power consumption of the steady state where only the PIR is active differs more than one order of magnitude with respect to a fully active state. This computational model is referred to as *Event-Driven Computational Model* since *the system is able to adapt the sub-systems activation based the context activity*. Similarly to this concept, other systems proposed a low-power secondary image pipeline for the early-detection of interesting events. *Mesheye* [51] integrates a power-hungry VGA image sensor that is activated based on the motion captured from a stereo low-resolution camera. To this scope, the platform includes two 6-bit grayscale 30x30 *Agilent ADNS-3060* sensor and a 32-bit *Atmel AT91SAM7S* MCU, which can be clocked up to 55MHz. The processing pipeline on the data coming from the stereo camera is composed of background subtraction and stereo matching. The higher resolution camera is activated if a positive match is determined by the visual processing. The reduced activation rate of the power-hungry camera enables a notable extension of the battery lifetime with respect to a scenario where it is activated very often. More recently, the *Glimpse* system [84] have presented a hardware-software flexible solution which includes a specialized imaging subsystem for always-on sensing. This latter comprises of a low-power MCU and an FPGA for running coarse vision algorithms. A filter rejection cascade is implemented on the platform for discarding uninteresting frames and therefore to realize a frame-selection classifier. Glimpse is able to detect frames representing events of interest over 87% (100%) of the time for visual events longer than 1s (3s) while drawing roughly 41-54mW for frame selection.

To exploit the opportunity raised by the mixed-signal processing, the smart camera *Wi-FLIP* [40] contains an imager with the focal-plane processing capabilities [37]. The digital processing unit is a *Marvell PXA271 XScale*, whose frequency can vary from 13MHz up to 416MHz with dynamic voltage scaling. This block is the energy bottleneck of the system, because the imager weights only for the 5% of the overall power cost (above 100mW when running at the minimum frequency). Hence, despite the available computational power and the proven functionalities, the power consumption of this system still exceeds the targeted consumption for always-on IoT applications due to the not-optimized processing platform. A more optimized smart camera design is presented in [16]. This system contains a vision chip with a large array of analog processing elements and a low-power FPGA, which is placed between a digital MCU and the sensor. The MCU is put in deep-sleep mode during the light integration period. The FPGA, which is fed by a 32kHz RTC, eventually triggers the MCU's wake-up for image readout and processing. By exploiting the sleep power mode and duty cycling, which results to be limited by the power supply and oscillator start-up times, authors report an average power consumption as low as 5.5mW within a surveillance application. The system proposed by Gasparini et al. [43] further exploit the event-driven model and focal plane processing by proposing an FPGA-based smart camera featuring the binary vision chip [47]. The FPGA design leverages two separate clock domains: a 15kHz low-speed clock drives the image sensing operation while a 15MHz high-speed clock is enabled for the data processing after the detection of a relevant motion on the focal plane. The presented processing pipeline aims at counting single persons passing in front of the camera. Thanks to ultra-low power cost of the imager and the power optimized design, the system presents a power consumption of 2.5mW when no motion is detected, while raises up to 4.2mW when fully active. But, despite that, the system lacks flexibility due to the hard-wired data processing, which is also limited to a single moving object detection. Recently, Kim et al. [57] presented an ultra-low-power smart camera

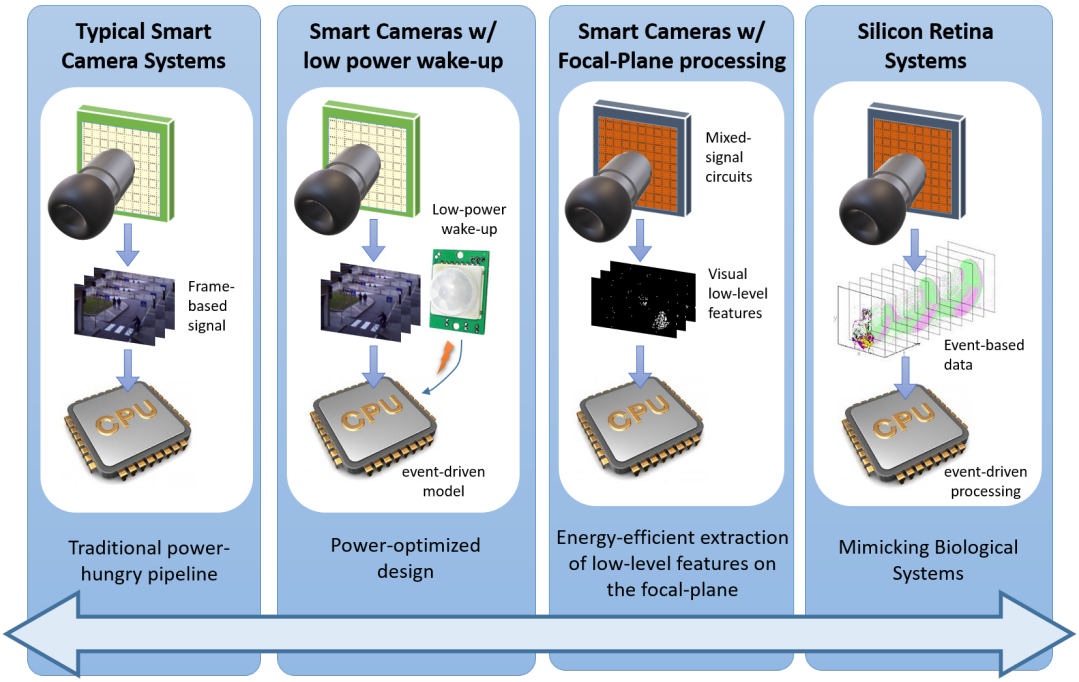


FIGURE 2.3: Diagram of energy-efficient imaging technologies [Silicon Retina system - Credit: https://github.com/robotology/event-driven].

that couples mixed signal and Event Driven digital processing on the same chip. The imager includes an analog processing unit, implementing the Viola-Jones detector, to filter out "non-face" images. To enable multi-scale face recognition, image analog downsampling is obtained by a switched capacitor circuit. By filtering the signal before the A/D conversion, a 39% power reduction is achieved. If candidate points are not discarded by early-filtering, a digital processing circuit is activated to refine the face detection based on the digitally-converted data. The digital circuit consists of an integral image generator and a Haar-like filtering unit. Measurement shows an average power dissipation of $96\mu W$ if a positive sample is identified by the analog pre-processing when running at 1fps, while for "non-face" conditions the power reduces as low as 24 μW .

2.4 Neuromorphic Visual Sensing and Processing

Biology has always been a source of inspiration for building intelligent and efficient systems [95]. In the late 1980s, Carver Mead opened the way of Neuromorphic Computing, which aims at mimicking neural architectures by means of analog, digital and mixed-signal circuits or software stacks [79]. The challenging goal comprises understanding and replicating the complexity of biological systems to increase the energy-efficiency of artificial intelligent agents (as a reference the human brain is estimated to consume less than 20W [34]).

In recent times, several research groups in the field of neuromorphic engineering are actively pursuing different approaches towards the emulation of neuronal systems, either for sensing or processing purposes. A biological system is generally modeled as a large network of neurons connected through synaptic links, through

which signals flow asynchronously [106]. Every neuron acts as a small computational core by applying non-linear filtering on the input signals, which can be amplified or attenuated by the prior synaptic connections. Given the high amount of neurons contained within a network, a biological architecture features a massive level of parallelism and distributed computation. The flowing signal and the carried information is typically coded in a spike-based fashion to emulate the neural impulses circulating within a nervous system. To efficiently handle the processing of this spike-based data, novel neuromorphic architectures support a data-driven computation, also referred as *Event-Driven*, and feature distributed and extremely parallel non von-Neumann architectures [80].

In the context of visual sensing, researchers have also pushed new imaging technologies to mimic the behavior of the human eye [31]. Likewise the biological counterpart, a silicon retina features local processing and gain control, at the pixel-level, to produce an asynchronous stream of digital spikes. With respect to a traditional CMOS camera, in a silicon retina each individual pixel produces asynchronous events based on the context dynamics. Hence, the digital output of a pixel circuit is a binary signal that is raised to '1' to signify the event detection. Within this imaging technology, the visual signal loses completely the meaning of frame. Such output is rather driven by the individual pixel activity and moves towards a *frame-free* flavour [30]. **Despite a more diffused time-driven readout scheme, where a host system periodically polls the sensor for reading the sampled image frame, silicon retinas, also named Event-Based Cameras, act as a master in the communication: whenever a single event is detected on the focal plane, the sensor decides to trigger a pulsed signal, which is indicated as an *Event*.** This is completely in contrast with traditional cameras, which produce sequences of highly redundant image frames.

To dispatch the generated events, *Event-Based Cameras* employ a de-facto standard encoding, namely the Address Event Representation (AER). According to this scheme, any spike information is issued asynchronously by the retina output interface by sending the (x,y) coordinates of the firing neuron within the 2D array of the retina, i.e. the pixel coordinate. An optional information attached to the address is the type of the event, needed if the retina features multiple kinds of spiking processes. The majority of the silicon retinas, which can be found in the literature, feature the AER scheme to encode the generated events [70, 68, 90, 15, 69, 103, 12]. Among the presented designs, the *Dynamic Vision Sensor* (*DVS*) by Lichtsteiner et al. [70] features two types of asynchronous events, namely ON and OFF Address Events (AEs), to signal positive and negative scene reflectance changes. Any photoreceptor circuit monitors the log-intensity change of the pixel voltage since the emission of the last event. Once the log-intensity change exceeds a threshold value, an ON or OFF event is emitted depending if the voltage level is increasing or decreasing. A DVS camera with higher sensitivity has been shown by adding more gain and bandwidth to the photoreceptor that precedes the differencing amplifier [68]. The *ATIS* sample by Posh et al. [90] integrates into the pixel circuit a temporal DVS sub-pixel to trigger a time-based intensity readings in a second subpixel. Doing this, the ATIS imager features an event-triggered and wide dynamic range intensity readout at the price of a larger pixel size and small fill factor. More recently, the *DAVIS* imager was proposed to generate either asynchronous brightness-change events and synchronous intensity values [15]. The sensor showed a power consumption between 5mW to 14mW depending on the DVS activity and not including the ADC converter.

Thanks to the low-latency of the sensing process, several applications employing event-based cameras have been documented within the robotic [82, 32], urban

monitoring [74] and gesture recognition domains [67]. To extract high-level features and information from the spiking data, some *Event-Driven Processing* techniques can be applied. So far, object tracking has represented one of the principal targets for the event-driven computation [74, 67, 64]. To track moving objects, an event-based camera is placed in a fixed position, i.e. it is not moving while capturing the visual signal, and therefore the illumination-change events can be attributed to moving objects on the field of view. In this context, the tracking models can be updated based on the spatio-temporal relation of the generated events, which are processed as soon as they get triggered. Some of the reported algorithms exploit a temporal clustering process to group together the visual events [74, 30]. More recently, novel mathematical models have been developed to build more complex tracking methods [64, 42]. In addition to these, other approaches focus on event-triggered convolutional processing [97], which are still in a raw shape but a rapid growth is expected in the next years thanks to the increasing number of research groups investigating in this area.

In the perspective of building ultra-low-power smart cameras, the innovative neuromorphic technologies and the event-based paradigm have become extremely relevant. From a system-level viewpoint, the following features result extremely beneficial to reduce the energy consumption of current imaging technologies.

- *A silicon retina dispatches only significant information to a host system, in the form of events*. The event generation process is enabled by focal-plane processing circuits integrated at the pixel level and it is triggered in response to varying lighting conditions in the camera field of view, also due to motion. A silicon retina outputs a compressed AER information, therefore **extremely reducing the sensor-to-processor bandwidth** of traditional imagers, which, on the contrary, produce a stream of highly redundant data frames.

- *The (digital) post-processing action is triggered by the event reception.* In absence of any relevant occurrence (e.g. due to not moving objects) in the retina field of view, a processing platform can be kept in idle mode, with low-power dissipation. Instead, when the event-rate increases, event-based data can be transferred to the processing unit for data analytics. Therefore, data processing is applied on non-redundant data, potentially **demanding for lower computational cost** than traditional frame-driven computation.

The aforementioned motivations have started to be explored to build smart camera nodes. The event-driven camera system proposed by Texeira et al. [104], named *XYZ*, is composed by a μW event-based imager [103] and a digital processor *OKI ML67Q5002* that features an ARM7TDMI core running at 58MHz and a wide variety of peripherals. Authors showed a lightweight event-based processing for aggregating imager data by running basic recognition tasks. However, not any optimization has been investigated to minimize the system power consumption. The camera-based system in [75] also makes use of an embedded processor, a *Blackfin DSP BF537* from *Analog Device* with a maximum frequency of 600MHz, to implement an event-driven processing for visual tracking based on data coming from the DVS sensor [70]. An external 32MB FIFO memory is placed between the imager and the processor for buffering the asynchronous events. Still, the power consumption results to be up to hundreds of mW due to the not-optimized system architecture and the power-hungry processing system. In contrast, the event-based tracking system *eDVS* [83] can consume less than $200mW$. This is composed of a DVS chip connected to a 64MHz 32bit MCU (*NXP LPC 2106/01*) with 128KB program flash memory and 64KB SRAM, which enables event-driven local processing and successfully serves for tracking applications.

2.5 Summary

This Chapter highlighted the issues and the current trends concerning the design process of power-optimized smart camera systems. A diagram with the available technologies for smart camera systems is depicted in Figure 2.3.

Traditional design approaches show energy-inefficient architectures due to power hungry data processing units, mostly comprising of mid-to-high end CPU processing platforms, and redundant image sensors. Despite the proven data analytic capabilities and flexibility, these architectures show a power cost of several hundreds of mW, which is not suitable for battery powered devices.

To address the energetic issues, novel techniques have recently emerged, also inspired by biological systems.

- Focal-plane processing is enabled by integrating analog processing circuits on the sensor die. When embedding these capabilities, an image sensor can perform the early-extraction of low-level visual features before the A/D conversion. This reduces the computational workload of digital post-processing actions. Moreover, the power consumption can be extremely reduced with respect to traditional image sensors, thanks to a power-efficient analog subsystem.

- Neuromorphic approaches for sensing and processing have demonstrated a viable way for reducing either the sensor-to-processor bandwidth and the visual processing workload by means of Event-Driven computation. According to it, data analytics is performed on relevant events while the system can be kept in idle mode if no interesting information is early-detected. Such paradigm motivates also the usage of low-end processing units for data analytics, to eventually reduce the overall energy cost of ultra-low-power smart cameras.

Chapter 3

Event-Based Sensing and Processing for Ultra-Low-Power Vision

3.1 Overview

Energy-efficiency is a key metric when addressing the design of autonomous visual sensing devices for the IoT ecosystem. Any choice made at the design time can severely affect the energy consumption of the node, potentially provoking a rapid battery discharge. When targeting a low-energy consumption, the design flow must trade-off between power costs and accuracies of the running computer vision tasks, as discussed in Chapter 1. This is intuitively clear by considering an aggressive duty-cycled approach. When reducing the sampling rate of the sensor (e.g. the frame rate), the average power of the system consumption can be reduced as low as the sleep power. But, this comes at the cost of a much longer latency and lower reactivity. Potentially, some relevant data can be lost by means of such a simple strategy, leading to a degraded sensing quality and performance.

To meet the energy requirement of IoT battery-powered smart cameras, a power highly-optimized design process is required, either concerning the sensing and the processing sides. Current off-the-shelf smart visual systems feature a power consumption in the range of tens to hundreds of mW, which is far from the μW target needed to guarantee a battery lifetime in the order of years. A smart camera system architecture include a CMOS imager (e.g. [87]), that produces a stream of image frames at a fixed frame-rate, and an embedded processing unit, which reads data from the sensor and runs computer vision algorithms on every image frame. This execution flow, also referred to as *frame-driven* computational model, consists of repetitive actions, namely sensing, transfer and processing, which can be redundant within some applications scenarios. Therefore, to address the power issues of traditional systems, a rethinking of the image acquisition and processing pipelines becomes a need, comprising both of hardware and software complementary design.

Concerning the sensing action, potential benefits for ultra-low-power vision can arise from novel focal-plane processing techniques, introduced in Section 2.2. According to this approach, some mixed-signal processing circuits can be integrated on the focal plane of the image sensors to (a) lower the sensor bandwidth and (b) perform a first filtering task in an efficient way. Thanks to the in-sensor processing, the power-hungry A/D conversion circuit has to operate on a lower amount of data and therefore the average power cost of the imaging task can be reduced if compared with traditional imaging pipelines. The potential benefits of focal-plane processing when serving for event-based sensing will be investigated in this Chapter.

As introduced in Section 2.4, in recent years neuromorphic engineers have considerably pushed the state-of-the-art of bio-inspired hardware circuits. In the context of visual sensing, silicon retinas have been developed to study and emulate the efficiency of biological sensory systems. These imagers feature an intelligent pixel design, which makes use of integrated mixed-signal processing circuits to enable pixel-level detection of relevant information. The sensor output signal is a stream of *events*, which are asynchronous dispatched whenever a light change is triggered by the pixel photodetectors. The generated events are encoded with an Address-Event Representation (AER) scheme , which consists on dispatching out the (x,y) coordinates of the firing neuron within the image plane. Image sensors featuring the AER readout scheme are also referred to as *Event-Based Cameras*.

From a system-level viewpoint, the event-based paradigm provides relevant energy saving opportunities. As a first observation, the sensor-to-processor bandwidth depends on the context activity. Since a silicon retina responds to changing stimuli in the camera field of view, any event will not be generated when observing a fixed scene and therefore the energy is not wasted for transferring useless or redundant data. Potentially, the datarate is close to zero when observing a static background. The second observation concerns the event-driven computation introduced in Section 2.4. As the digital engine is fed by a set of non-redundant visual information, the workload of the digital processing will be lower than if applying data analytics on raw frame-based data. Hence, another energy saving contribution with respect to tradition visual sensors is related to the lower computational complexity of visual task when dealing with event-based data input.

The Chapter will discuss the hardware-software design process and the energy benefits of exploiting event-based sensing, focal-plane processing and event-driven computation for ultra-low power visual sensing. To demonstrate the design approach, this Chapter introduce a visual system architecture coupling an ultra-low-power event-based camera with a programmable quad-core digital processor, which is optimized for high-energy efficiency. The processing unit enables a local event-driven processing on data coming from the sensor. To demonstrate the advantages of the event-based sensing and processing, the presented architecture is benchmarked against a visual monitoring application. An object tracking flow, derived from a neuromorphic algorithm for tracking moving objects, is implemented and adapted to efficiently run on a quad-core processor. Based on this implementation, an energy estimation is conducted along with a comparison with respect to traditional frame-based camera systems.

Summarizing, this Chapter includes:

1. The HW-SW definition of an event-based smart camera system architecture, featuring ultra-low power consumption.

2. The implementation of an event-driven algorithm for moving objects detection, which is optimized to run efficiently on a parallel embedded processor.

3. The detailed quantification of the energy efficiency improvements of the proposed event-based system with respect to a traditional frame-based vision flow.

The remainder of this Chapter is organized as follows. Section 3.2 provides a detailed description of the system architecture, either regarding the event-based camera and the processor. An event-driven algorithm is described in Section 3.3,

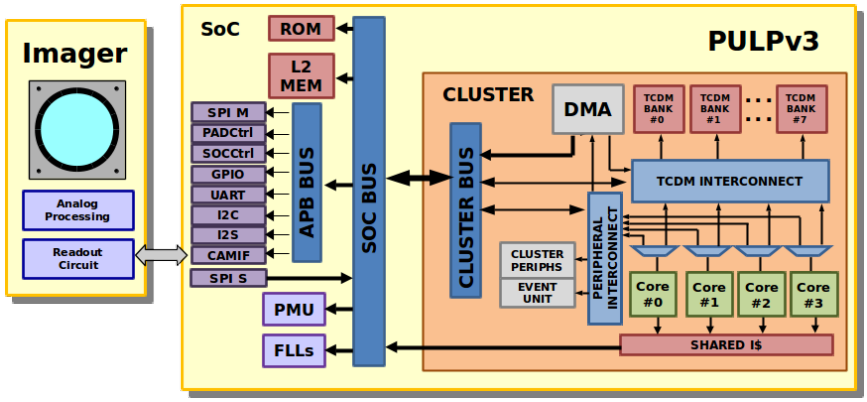


FIGURE 3.1: Event-driven smart camera system architecture.

while Section 3.4 reports the implementation details on a 4-core platform. The implementation is optimized to run faster on the selected platform, and at the same time, reduce the energy consumption of the processing task. Section 3.5 contains the experimental analysis of the proposed architecture conducted against a monitoring application dataset. Lastly, Section 3.6 summarizes the main results of the Chapter.

3.2 Event-Driven Smart Camera System

To demonstrate the advantages of event-based sensing and processing for Ultra-Low Power (ULP) visual sensing, this work considers a node architecture including a ULP 128x64 spatial contrast binary imager [47] and a fully programmable 4-cores ULP platform (PULP [96]). This architecture combines the mixed-signal focal-plane processing of the imager, which is aimed at producing visually relevant events, with the fully programmable parallel digital processing. The imager internally performs pixel-level contrast extraction, binarization and temporal frame-differencing, and produces address-event coded information, namely the *Events*, while consuming $100\mu W$ at 50fps. The processing platform processes the arrays of visual events produced by the imager to extract high-level information for a power cost of $2.9mW$ at the frequency of $80MHz$, and supply voltage V_{DD} of 0.55V. This section details both the sensing and processing devices, providing useful insights of their implementation. A block diagram of the system is reported in Figure 3.1.

3.2.1 Ultra-Low-Power Event-Based Camera

The event-based imager developed by Gottardi et al. [47] integrates pixel-wise mixed-signal circuits on the focal plane to compute the spatial-contrast among neighboring pixels. For any pixel location *PO*, the local gradient is computed with respect to two adjacent pixels *PN* and *PE*. The spatial-gradient kernel is illustrated in Fig. 3.2a. Assuming *PO* and *PN* to be respectively the more and the less exposed pixels to light, the gradient will be proportional to the voltage difference between *PO* and *PN* (fig. 3.2b). Figure 3.2c illustrates the circuit that implements this pixel-wise functionality. The *Contrast Block* transduces the spatial gradient into a voltage level. As soon the most illuminated *PO* voltage value crosses the V_q threshold, the comparator *comp1* switches the output value and activates the voltage linear integration within the *Contrast Block*. The integration stops when the less illuminated pixel voltage *PN*

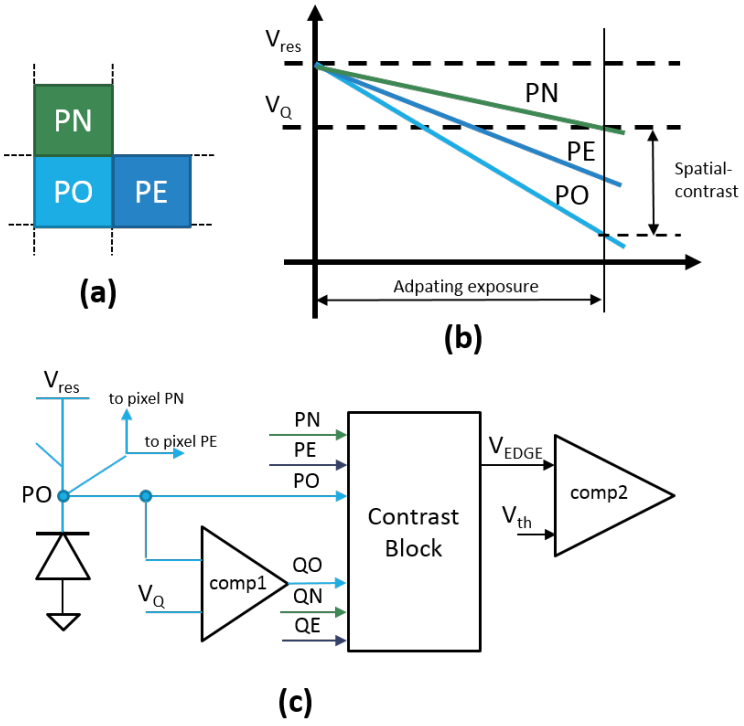


FIGURE 3.2: (a) In-sensor pixel mask to compute the binary gradient (b) Gradient extraction approach (c) Mixed-signal circuit for contrast extraction that is placed at every pixel location.

goes above the V_q threshold. This implies a pixel-level self-exposure capabilities. Then the output of the contrast block is binarized by means of the comparator *comp2* with respect to a tunable level V_{th}. On the adopted prototype, V_q is generally tuned between 2.7V and 3V with a minimal impact on the output image. On the contrary, V_{th} is recommended to be tied to 0V to gain a higher sensitivity. After the binarization, a 1-bit analog buffer can store the binary pixel value of the latest acquisition. If requested, the frame difference between the current binary image and the one stored can be dispatched as an output by the sensor.

If looking at the supported analog processing capabilities, the mixed-signal circuit enables the following operations: (a) *contrast-extraction*, by directly dispatching the binarized contrast value, (b) *background subtraction*, by differencing with respect to an initial stored sample, (c) *motion-detection*, by performing the frame-difference with respect to the latest frame. It is worth noting that, in this latter scenario, only the changing contrasts will generate a non-zero value. When the camera is placed in a fixed position, the motion-detection mode will generate a binarized information describing the motion activity in the field of view of the sensor.

The image sensor only readouts the asserted pixels of the binary frames, i.e. the pixels with a non-zero value after binarization and frame-difference. An AER coding scheme is employed to keep the bandwidth low in case of a sparse output, as it is generally the case especially when in motion-detection mode. Therefore, the sensor output is formatted as a stream of *Events*, each one described by its (x,y) coordinate and the (optional) sign after frame difference (to distinguish between the $1-0$ and $0-1$ cases). Thanks to this feature, the sensor operates as an *Event-Based*

camera but still preserving a discrete timing for the generation of the event-based information. Differently from silicon retinas of Sec. 2.4, the *events* detected within the image plane are readout according to a raster scan order after an exposure period, instead of being asynchronously dispatched. Nevertheless, the amount of transferred data is varying depending on the context-activity, following the event-based sensing paradigm.

To save energy, the sensor features two different readout modes. In *Idle* mode, the asserted pixels (i.e. pixels with non-zero values after the binary frame difference) are internally counted and only the counter value is provided as an output. Therefore, no data transfer is occurring within this operating mode. In *Active* mode, instead, the event-based data are dispatched out by means of a raster-scan process. The average power consumption is extremely affected by the readout mode because of the expensive pad activity that can be saved in *Idle* mode. Indeed, the measurements reported in [47] show a power consumption up to $40\mu W$ and $100\mu W$ when respectively in *Idle* and *Active* mode (at a frame rate of 60fps and a 25% pixel activity). The power scales linearly with respect to both the pixel activity and the frame-rate. As a remark, the ultra-low-power consumption is due both to (a) the mixed-signal focal-plane processing and (b) the event-based readout coding.

3.2.2 Parallel Platform

PULP (Parallel Ultra Low Power) is a multicore processing system targeting high-energy efficiency to satisfy the computational requirements of a wide range of applications constrained by power budgets of few *mW* [96]. The *PULPv3* System on Chip (SoC) includes a 4-core cluster and several IO peripherals. A 28nm FD-SOI chip prototype has been designed and fabricated. The SoC architecture is depicted in Fig. 3.1.

The compute engine of the *PULPv3* architecture is a cluster with 4 cores. The processor micro-architecture is based on the *OpenRISC Instruction Set Architecture* (ISA), which has been extended to support energy efficient DSP operations, such as *Single Instruction Multiple Data* (SIMD) instructions. Moreover, the extended ISA features zero-overhead hardware loops with L0 I-buffer, load and store operations embedding pointer arithmetic and power management instructions [44]. In addition to the processing cores, the *Cluster Region* includes a 48kBytes multi-banked *Tighly Coupled Data Memory* (TCDM) working as software-managed L1 scratchpad memory, avoiding memory coherency overhead of data cache. The TCDM features 8 word-level interleaved banks connected to the processors through a non-blocking interconnect to minimize banking conflicts, and a Direct Memory Access (DMA) engine to handle data transfer with L2 memory.. The cores share 4Kb of instruction cache with support for broadcast to exploit the SIMD behavior of several signal processing algorithms, further increasing energy efficiency [76]. Among the cluster peripherals, the *Event Unit* autonomously handles the clock-gating of individual cores in idle state (e.g. if waiting at a synchronization barrier).

The off-cluster region, also named *SoC Region*, contains a 64kBytes L2 memory and the peripheral subsystem, which includes several IO interfaces, such as SPI, UART and I2C. Additionally, a specialized camera interface transfers data from the imager to the L2 memory. The internal architecture of this module will be extensively discussed in Chapter 3.

On this platform, an optimal power management model can be obtained by means of *Dynamic Voltage and Frequency Scaling* (DVFS) techniques. To enable it, the Cluster and the SoC Regions feature different power and clock domains. The

clock frequencies are internally generated thanks to dedicated *Frequency Locked Loops* (FLLs), which are placed in the so-called *FLL region*. This latter is powered at 1V. The FLLs are fed by a 32kHz external clock oscillator and allow a fine-grain configuration.

PULP is a fully programmable platform leveraging C and OpenMP for software programming. This leads to a complete flexibility for application development and eases the design of the signal-processing flows.

3.3 Event-Driven Processing

Tracking moving objects is a relevant application domain for event-based cameras [74, 30]. When a camera is placed in a staring configuration, changing temporal or spatial gradients are detected in correspondence of object movements. In this context, an event-based camera produces a sparse AER signal, whose bandwidth, i.e. the event rate, varies depending on the context activity. As a simple observation, more moving objects in the camera field of view will cause a higher event rate, while a static background will potentially imply a near-zero event generation rate.

Litzenberger et at. [74] presented an event-driven tracking algorithm by applying a clustering procedure on the visual events detected when moving objects appear on the image plane. The proposed algorithm relies on a distance metric, considering both the spatial and temporal domains. The output is a list of N *blobs* B_i, $i = 1, ..N$, also referred as *tracker*, each one corresponding to a tracked object. Any detected blob is characterized by a center of mass $C_i = (c_x, c_y)$, a radius R_i (in case of circular blobs) and a weights W_i. When an *event* $p = (p_x, p_y)$ is triggered, it is assigned to the closest tracker, determining an update of the blob list. To this aim, a *Seek Region* is also defined to indicate the maximum distance for an assignment process. In case of circular blobs, a radius S_i describes the seek region. Therefore an event p is assigned to one of the tracker B_i belonging to the blob list, labelled as $b \in \{1, N\}$, such as:

$$\begin{aligned} b = \quad & \arg\min_i |p - C_i| \qquad i = 1, ..N \\ \text{s.t.} \quad & |p - C_i| \leq S_i \end{aligned} \tag{3.1}$$

After the assignment, the blob's characteristics C_b, R_b, S_b and W_b are updated according to the following rules.

$$C_b = \alpha \cdot C_b + p \cdot (1 - \alpha) \tag{3.2a}$$
$$R_b = \min(R_{min}, \quad \alpha \cdot R_b + |p - C_b| \cdot (1 - \alpha)) \tag{3.2b}$$
$$S_b = \min(S_{max}, R_b \cdot m) \qquad 1 < m < 3 \tag{3.2c}$$
$$W_b = W_b \cdot \alpha + 1/dt \cdot (1 - \alpha) \tag{3.2d}$$

where $0 < \alpha < 1$, usually set near 1 for smooth tracking, and dt is the timestamp difference between the current and the last event reception. R_{min} and S_{max} constraint the blob size and the seek distance to be kept within certain limits. If not any existing tracker matches the condition 3.1, a new blob is initialized. The blob list is periodically scanned to delete inactive trackers or for merging distinct items belonging to the same moving objects. This event-driven tracking approach has been used to control a robotic goalie, achieving an an effective frame rate of 550 FPS and a reaction latency of 3ms with a 4% processor load using standard USB interfaces [32]. The algorithm has been also extended to feature blobs with multiple shapes and

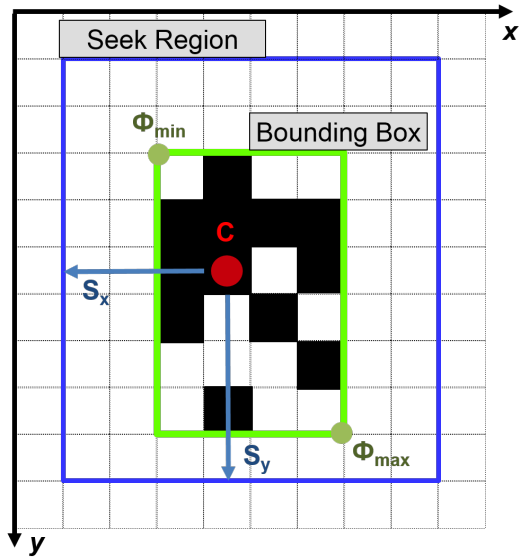


FIGURE 3.3: Blob features extracted through the event-driven processing.

orientations, by considering Gaussian, Gabor, combinations of Gabor functions, and arbitrary user-defined kernels at a cost of a higher computation complexity [64].

In this work, the tracking algorithm described above is adapted to be applied on the data generated by the imager of Section 3.2.1, when configured in motion-detection mode. Such image sensor generates an array of AER events with a discrete timing and dispatches them in a raster scan order. Every pixel can trigger only a single event for a given time slot (i.e. the frame period). Following the flow of the event-driven tracking algorithm described before, a list of blobs is computed by analyzing the array of N_{EV} events $p = (p_x, p_y)$ produced at a given frame time and sorted by means of the raster-scan readout. It is worth to note that the amount of data N_{EV} depends on the motion detected in the context. The tracking is performed by clustering the events based on a distance criterion and consists of the following three steps:

1. *Blob Formation*, in which the events are clustered to form blob tracking structures, based on a distance metric and given the list of existing blobs.
2. *Filtering*, in which the blobs containing a low amount of events are deleted.
3. *Merging*, in which multiple blobs that are close to each other are merged together.

The extracted blobs B_i, $i = 1,..N$ feature a rectangular shape and are described by the following characteristics: a center of mass $C_i = (C_{x_i}, C_{y_i})$, the number of events W_i, the corners of the rectangular bounding box, ϕ_{max_i} and ϕ_{min_i}, and the seek region distance $S_i = (S_{x_i}, S_{y_i})$. Figure 3.3 graphically illustrates the blob features. Note that C_i, ϕ_{max_i}, ϕ_{min_i} and S_i are two dimensional vectors, described by the x and y components.

The *blob formation* is performed as follows. Let N be the number existing trackers and let C_i and S_i, $i = 1,..N$, respectively, be their centers of mass and their seek distances, defined as before. Additionally, the 2D vector $\Omega_i = (\sum p_x, \sum p_y)$ accumulates the event coordinates sums along both axes.

1. Each of the N_{EV} events $p = (p_x, p_y)$, is assigned to one of the existing blob, with index b, which satisfy the following criteria.

$$\begin{aligned} b = \quad & \arg\min_i d_{L_1}(p, C_i) \qquad i = 1, ..N \\ \text{s.t.} \quad & |p_k - c_{i_k}| \leq S_{i_k}, \quad k \in \{x, y\} \end{aligned} \tag{3.3}$$

where $d_{L_1}(\cdot)$ is the L_1 distance. If such a blob exists, its descriptors are updated as follows:

$$\Omega = \Omega + p \tag{3.4a}$$
$$W = W + 1 \tag{3.4b}$$
$$\phi_{max} = \max(\phi_{max}, p) \tag{3.4c}$$
$$\phi_{min} = \min(\phi_{min}, p) \tag{3.4d}$$

where, W_i and Ω_i are initialized to 0. Once the whole set of N_{EV} events is scanned, the center of mass of the blobs in the final list can be computed as $C_i = \Omega_i / W_i$.

2. Events not assigned in the previous step seed a new set of blobs, stored in a separate list. For notation purpose, the new blobs, along with their features, are marked with the symbol $\sim$. Hence, $\tilde{B}$ is the list of new $\tilde{N}$ blobs. The first unassigned event instantiates a new blob, whose center of mass corresponds to the event coordinates. Its seek distance $\tilde{S}$ distance is setup to be (δ, δ), where δ is a user-defined parameter. Let $\tilde{N}$ be the number of new blobs detected at a given point. A pixel p is assigned to one of them such that:

$$\begin{aligned} b = \quad & \arg\min_i d_{L_1}(p, \frac{\tilde{\Omega}_i}{\tilde{W}_i}) \qquad i = 1, ..\tilde{N} \\ \text{s.t.} \quad & |p_k - \frac{\tilde{\Omega}_{i_k}}{\tilde{W}_{i_k}}| \leq \tilde{S}_{i_k}, \quad k \in \{x, y\} \end{aligned} \tag{3.5}$$

In this equation $\tilde{\Omega}$, $\tilde{W}$ and $\tilde{S}$ are the new blobs descriptors, which are updated after any pixel assignment according to (3.4a-d) and:

$$\tilde{S} = \min((\tilde{\phi}_{max} - \tilde{\phi}_{min})/2 + \delta, \quad R_{MAX}) \tag{3.6}$$

where R_{MAX} has been introduced to limit the increment of the seek region distance and, hence, of the blobs size. This parameter is tuned according to the dimension of the object to be detected. To save computation resources, an approximation is introduce in equation 3.6, such as $(\tilde{\phi}_{max} - \tilde{\phi}_{min})/2 \approx (\tilde{\phi}_{max} - \tilde{\phi}_{min}) \gg 1$.

After formation, the *Filtering* phase aims at removing from the lists B and $\tilde{B}$ any blob formed by a low number of events. The resultant filtered items are collected back in the original blob list B.

Eventually, a *Merging* phase produce a final blob list by merging together blobs that are close to each other or even overlap. The merging operation between blobs B_i and B_j takes place if:

$$|C_{i_k} - C_{j_k}| \leq S_{i_k} \quad or \quad |C_{i_k} - C_{j_k}| \leq S_{j_k}, \quad k \in \{x, y\} \tag{3.7}$$

TABLE 3.1: Blob Descriptor Structure

Symbol	Description	Format	Bytes
C	Center of Mass	*2x16 vector*	4
S	Seek Region Distance	*2x16 vector*	4
ϕ_{max}	$\{\max(p_x), \max(p_y)\}$	*2x16 vector*	4
ϕ_{min}	$\{\min(p_x), \min(p_y)\}$	*2x16 vector*	4
R	$(\phi_{max} - \phi_{min}) >> 1$	*2x16 vector*	4
Ω	$\{\sum p_x, \sum p_y\}$	*2 unsigned int*	8
W	Number of events	*unsigned short*	2
$pntr$	Pointer to next item in the list	*unsigned int*	4

Eventually, another filtering pass is performed after merging to further refine the final blob list.

3.4 Implementation of an Event-Driven Object Tracking Algorithm on a 4-core Embedded Processor

This section describes the implementation of the Event-Driven algorithm on the parallel processor described in Section 3.2.2. Initially, the implementation targets a single core execution and then it is extended to a 4-cores processing.

The cores included within the PULP platform feature a OpenRISC extended ISA, which supports SIMD vector instructions. Due to the *(x,y)*-vector format, either the events triggered by the imager and the blob features (centers of mass, boundary coordinates of the bounding boxes, seek-region distances) can be mapped straightforwardly on 2x16 bit vectors. Table 3.1 details a stored descriptor for any blob item, along with the memory requirement. Every entry of the list occupies a total of 34 Bytes. Both the blob lists B and $\tilde{B}$ can contain a limited number of items, generally set as 16. If more blobs are detected, a new item will be instantiated by replacing one of the blobs with the lowest number of events. During the computation, the tracker structures are stored in L1 memory to reduce the memory access latency and therefore speed-up the execution. After completion, the output list is copied back to the L2 memory, due to the data retention property.

The algorithm implementation is optimized by benchmarking against the visual dataset that will be described in the next section. Within the implementation, the Eq. 3.5 is verified by avoiding any numerical division, which is expensive due to the absence of any HW dividers. Indeed, a more lightweight formulation is obtained by reversing the equation. Figure 3.4 summarizes the execution time of the event-driven processing after applying multiple optimization steps. The execution time is normalized with respect to the number of clock cycles required to run the video benchmarks on an ARM-Cortex M4 core (exploiting only 32-bit arithmetic instructions). On a single PULP core, the average execution time is reduced by 4%.

A reduction of 25% is achieved by fully exploiting the ISA vector extension (indicated with label *HW Ext* in the figure). The operations of the Event-Driven algorithm imply a computation on both the x- and y- coordinates, either for solving the minimization problems (Eg.3.3-3.5) and for updating the blob descriptor (Eg.3.4-3.6). Therefore, a relevant latency saving is achieved by means of instruction-level parallelism, enabled by 2x16 bit vector SIMD operations.

To exploit the full computational power of the PULPv3 4-core cluster, the computational load is distributed among the available cores, by means of thread-level

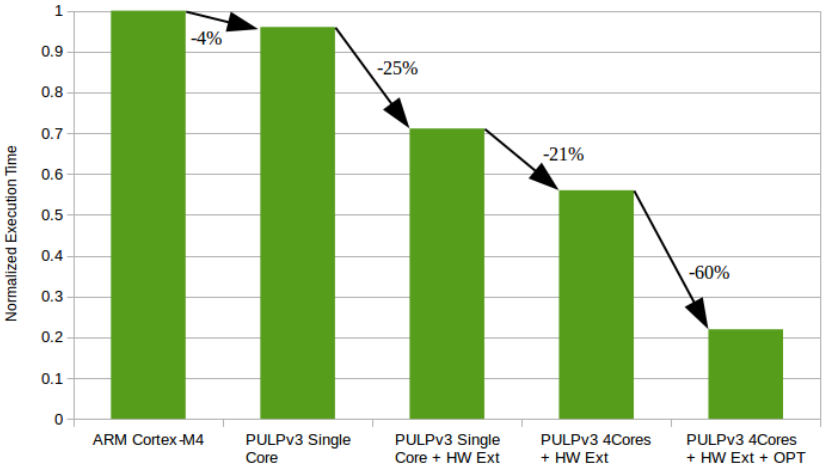


FIGURE 3.4: Comparison of software execution time (clock cycles) for different algorithm implementations. Values are normalized with respect to the execution time of the not-optimized algorithm running on an ARM Cortex-M4 core.

parallelism. To this aim, the input array of events is split into 4 separate sub-arrays and processed as described in Section 3.3. The *blob formation* phase represents the heaviest computational section of the entire algorithm (95% of the time on the collected visual dataset) and therefore benefits more from a parallel computation. The operations belonging to the step 1 of the *blob formation* procedure are highly parallelizable since the minimization problems can be solved independently by the different cores. Only a few write operations during the descriptors update are coded in a critical section to handle mutually exclusive access to the blob list B, that is instantiated as a shared variable. If an event cannot be assigned during this step, it will be processed for the formation of new blobs stored in the list $\tilde{B}$ (point 2 of the algorithm). To preserve the raster-scan order, the not-assigned events are marked within the first parallel scan and processed later by a single core. This fact represents a critical bottleneck for the parallel runtime. For instance, if the initial blob list B is empty, the *blob formation* phase needs to be entirely performed by a single core because the assignment condition of Eq. 3.3 will never be verified (N=0). For this reason, this first attempt of parallelization leads to a limited speed-up of 1.27x over the theoretical maximum of 4x, which is enabled by the available computational power of the 4 cores. More in details, the parallel runtime results to be intrinsically limited by Amdahl's law on this considered dataset. Indeed, only the 36.2% of the computation related to the *blob formation* phase is spent on the highly-parallelizable block (step 1 of the *Blob Formation* algorithm). Considering also a parallel filtering operation and a sequential scheme for the merging, a maximum speed-up of 1.4x is computed by applying the Amdahl's law, still far from the maximum 4x.

To overcome this limitation, the algorithm flow is modified, causing also some algorithmic drifts. Instead of marking the not-assigned events for later processing, every core handles a private list of new blobs which is filled by analyzing the data within the private input. Therefore, new blobs will be detected according to equations 3.5-3.6 based on the information contained in the private sub-array, instead of considering the whole sorted set of not-assigned events. The partial results will be independently filtered by the different cores and then collected together during the merging phase. A synchronization point is placed after filtering to wait for the

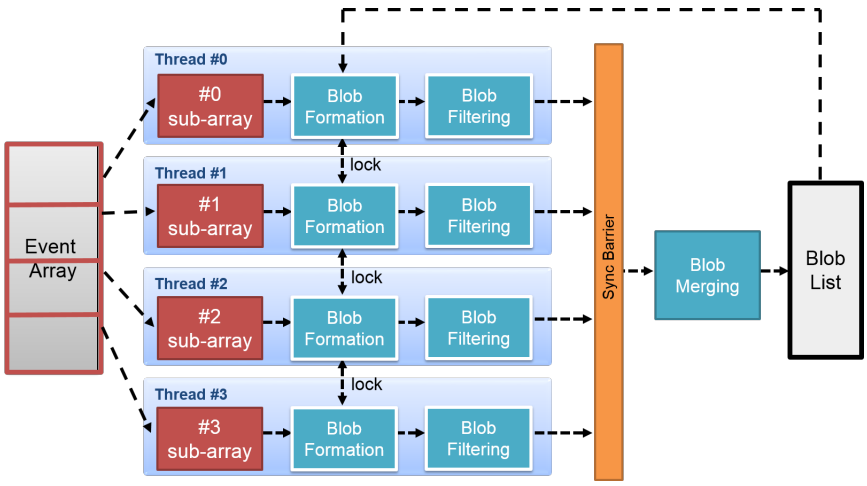
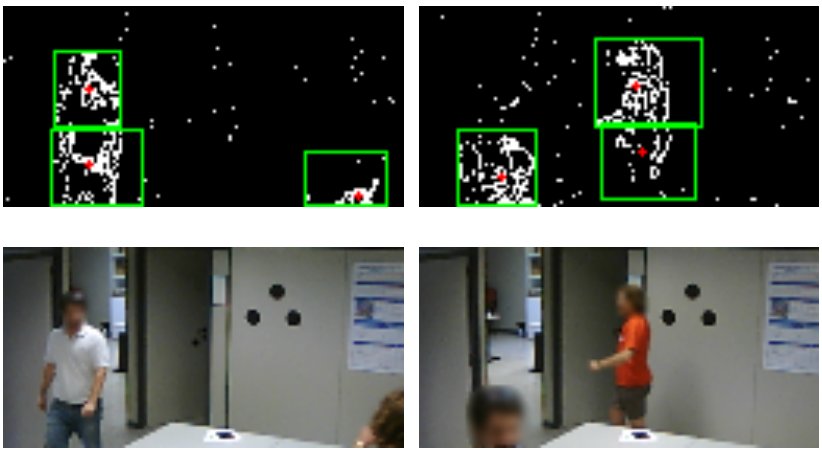


FIGURE 3.5: Block Diagram of the Optimized parallelization scheme.



(A) (B)

FIGURE 3.6: The pictures on top show two frames as captured by the imager. The detected blobs and their centers of mass are highlighted respectively in green and red. The corresponding ground-truth images are displayed on the bottom.

completion of the cores computation. The block diagram in Figure 3.5 illustrates the optimized parallelization scheme. Tests on the video dataset reveal that now 94% of the execution time is spent on the parallelizable section, which turns into a theoretical maximum speed-up of 3.39x according to Amdahl's law. Despite that, this optimized implementation (labelled with *OPT* in the figure) presents a slightly lower speed-up of 2.5x, due to (a) the unbalancing of threads concurrently running on the 4 cores (48% of the overhead) and to (b) the accesses to the critical section, parallelization overhead and contention in L1 memory. However, by running the optimized version on the 4-cores cluster, an execution time reduction of 60% is obtained with respect to that of not-optimized parallel implementation.

3.5 Evaluation

The proposed Event-Based Smart Visual System is evaluated on a real-life application consisting of tracking moving people within an indoor space. To this aim, an evaluation dataset is collected with the Event-Based Camera, which is placed in a staring configuration within an indoor room and configured in motion-detection mode. When collecting the dataset, the maximum integration time of the image sensor is set to 100msec. This determines an acquisition frame rate of 10fps, which is considered suitable for monitoring applications. Some examples of captured frames are shown in Figure 3.6. To test the proposed event-driven processing technique, six collected video sequences, each one composed of 340 frames, include moving people entering and exiting the camera field of view. In the entire dataset, the 36% of the frames show one moving person, the 3% show two moving persons, while the rest of the frames does not contain moving people. The evaluation of the event-driven algorithm implementation is conducted by gathering runtime statistics when the code is running on the PULPv3 cycle-accurate FPGA emulator. For each video, the object tracking starts by assuming that the blob list is initially empty.

Accuracy Evaluation. To quantify the accuracy and precision of the event-driven local processing, the following metrics are defined as follows and evaluated after manually extracting the bounding boxes of moving objects within the benchmarking videos.

$$accuracy = \frac{1}{|M|} \sum_{i \in M} \frac{|GT_i \cap BBX_i|}{|GT_i \cup BBX_i|} \tag{3.8}$$

where M denotes the set of frames which contain moving objects, $|M|$ is the cardinality of M, GT_i and BBX_i are the union sets of the bounding boxes of the ground-truth objects and of the detected blobs, respectively;

$$precision = \frac{n_{target}}{n_{target} + n_{fp}} \tag{3.9}$$

$$recall = \frac{n_{target}}{n_{target} + n_{fn}} \tag{3.10}$$

where n_{target}, n_{fp} and n_{fn} denote respectively the number of marked ground-truth objects, false positives and false negatives. On the collected video dataset, an accuracy of 0.70 and 0.71 respectively is obtained for the event-driven blob detection algorithm and its optimized parallel version. The precision achieved is 0.95 for the baseline low-parallelism algorithm, while the algorithm optimized for parallelism achieves 0.93. The recall is above 0.98 in both cases. Hence, the event-based approach is effective and its optimizations for increased efficiency do not compromise accuracy, precision and recall. An extended comparison among the event-based approach a traditional frame-based approach will be provided in next Chapter.

Local Event-Driven Data Processing. Table 3.2 contains some statistics after applying data processing on the benchmarking videos (labelled as *vid0-vid5*). The number of events per frame and consequently the imager bandwidth (denoted by *Avg Pixel* and *Imager BW*, respectively) depend on the context activity. In addition,

TABLE 3.2: Event-Driven Blob Detection Statistics

Benchmark Video	*vid0*	*vid1*	*vid2*	*vid3*	*vid4*	*vid5*
Avg Events (N_{EV}/ frame)	99.6	138.6	118.5	176.1	104.3	138.2
Imager BW [B/sec]	1992	2771	2370	3522	2085	2764
Avg MOPS	0.029	0.041	0.033	0.060	0.031	0.046
Peak MOPS	1.12	1.01	1.54	1.48	1.16	2.13
Avg Detected blob	0.45	0.70	0.68	1.17	0.60	0.77
System BW [B/sec]	144.0	224.0	217.6	374.4	192.0	246.4
BW Reduction	-92.8%	-91.9%	-90.8%	-89.4%	-90.8%	-91.1%

TABLE 3.3: Event-Driven vs Frame-Based Comparison

Approach	**Avg Pixels**	**Imager BW (KBps)**	**Avg MOPS**
Event Driven	129.2	2.52	0.040
Frame Based	8192	80	1.876
Gain	63.4x	31.7x	46.9x

the table reports the numbers of average and peak *MOPS*[1] needed to execute the algorithm and the number of detected blobs on each video sample averaged over the number of frames (*Avg Detected Blobs*). It is worth noting that the workload varies during the execution because of the variable input event rate. By considering a payload of 32 bytes to transmit externally a blob descriptor (the pointer feature is not necessary), the system bandwidth (*System BW*) results extremely lower (-90% lower) if compared to a streaming architecture that has to transmit the imager raw data (*Imager BW*).

In Table 3.3, the proposed event-driven system is compared with a traditional *frame-based* embedded vision system that uses an image sensor running at 10fps with 128x64 8bit pixels resolution. The data bandwidth generated by the traditional imager is $80KBps$, 31x higher than that of the event-based camera due to the address coding readout style. Moreover, a traditional system processes entire frames, hence the number of operations does not significantly vary frame by frame. For both the approaches, the table reports the mean number of operations per frame averaged over the videos (*Avg MOPS*) needed to perform the clustering of foreground pixels associated with moving objects. Within this evaluation, the frame-based baseline features a processing pipeline composed by several filters (frame difference, binarization, dilation and erosion) and the extraction of connected components.

Energy Evaluation. For the analysis of system energy costs, the PULPv3 cluster power model is applied, which is reported in Table 3.4. The cluster power densities, along with the maximum frequencies, are illustrated in Table for several V_{DD} voltages. The peak energy-efficiency is evaluated considering equivalent *OpenRISC* operations without ISA extension. The power and frequency figures reported in the Table are estimated with *Synopsys PrimeTime* on the post-layout database of the PULP cluster, which is implemented in 28nm *UTBB FD-SOI RVT* technology [41]. The 28nm *UTBB FD-SOI* libraries used for power and timing analysis are characterized for power supplies ranging from 0.5V to 1.0V in the typical corner case at

[1]Equivalent OpenRISC operations.

TABLE 3.4: PULPv3 Power Model

V_{DD} [V]	Max Frequency [MHz]	Dynamic Power Density [μW/MHz]	Leakage Power [μW]	Peak Energy Efficiency [GOPS/W]
0.5	55	19	41	301
0.6	119	27	44	218
0.7	238	37	66	159
0.8	366	42	100	141
0.9	480	53	150	110
1.0	498	78	231	76

TABLE 3.5: System Energy Costs Estimation and Comparison

Scenario	Frame Based	Event Driven			
	Camera [87] + STM32 [99]	Async Retina [12] + STM32 [99]	Imager[47] + STM32 [99]	Imager[47] + Apollo [6]	Imager [47] + PULPv3 [96]
Avg Ev/Frame	8192	130	130	130	130
Imager En $[\mu J/frame]$	**62.2**	**28.4**	**1.06**	**1.06**	**1.06**
Proc Clk Freq [MHz]	26	26	26	24	82
Proc Act Pow [mW]	8.6	8.6	8.6	2.7	2.9
Duty Cycle	72.1%	1.21%	1.21%	1.31%	0.11%
Proc En $[\mu J/frame]$	**623.7**	**10.82**	**10.82**	**3.67**	**0.73**
System En $[\mu J/frame]$	**685.9**	**39.22**	**11.88**	**4.73**	**1.79**

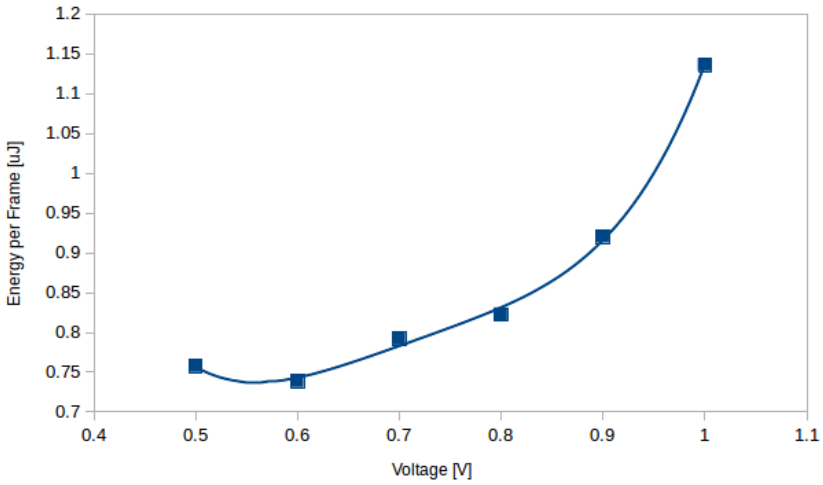


FIGURE 3.7: PULPv3 processing energy cost per frame on different operating points

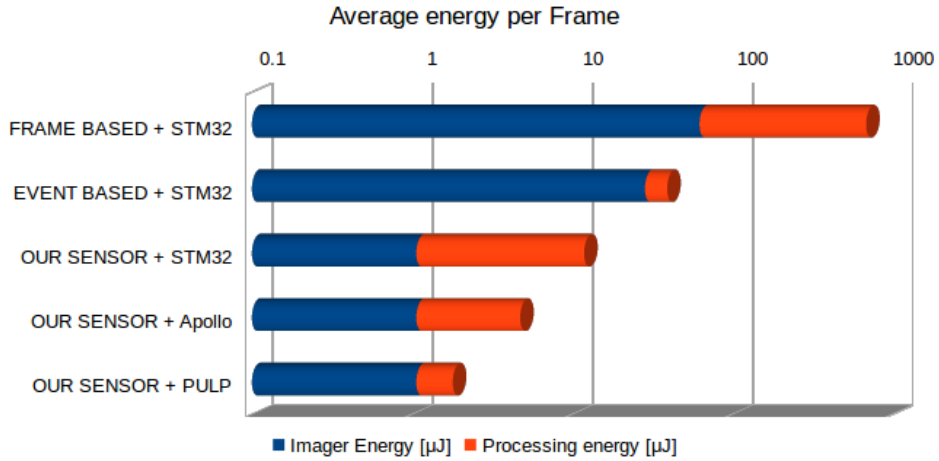


FIGURE 3.8: System Energy Cost Comparison

the temperature of 25C. The activity file (*.vcd*) used for the power analysis is extracted running a typical high-utilization workload. In addition to the cluster power consumption, an active power consumption of $1mW$ is estimated for the SoC, that includes L2, bus, clock and supply voltage generation and IOs.

To estimate the processing energy cost per frame during the tracking application, the event-driven execution is modeled as running to completion. Periodically, after imager readout, both SoC and cluster regions are enabled for data processing. When computation completes, the platform is put into deep sleep mode. The duty cycle is determined as the ratio between the event-driven execution time and the frame period (10msec). Figure 3.7 illustrates the processing energy cost per frame on several operating points (voltage and frequency). The minimum energy point is found for a cluster voltage of 0.56V ($V_{BB} = 0V$) and a maximum operating frequency of 82MHz. Given this operating frequency, the average application duty cycle results to be 0.11%, therefore the deep sleep power assumes a relevant role for energy budget requirements. On PULPv3 platform, by considering the leakage power of SoC, L2 memory and IO pads required to implement the protocol with the imager, the deep sleep power amounts to $4.2\mu W$.

Table 3.5 reports an energy comparison between the proposed system and other ultra-low power solutions. Power consumption figures related to the image sensor are measured on silicon samples [47] and scaled down according to sensor typical activity observed during the benchmark execution. The comparison baseline systems include a *Ambiq Apollo* processor [6], which features an ARM Cortex-M4F core and up to 64kBs RAM, and the off-the-shelf *STM32L476* [99], an ULP MCU with an *ARM Cortex-M4* core and 128kB SRAM. Among MCUs, *Ambiq Apollo* achieves the lowest power in sleep mode, as $0.33\mu W$, and the highest reported energy efficiency (8.6MOPS/mW). Its active power amounts to 2.77mW at maximum frequency 24MHz. On the other hand, the *STM32L476* is an energy-efficient MCU that consumes $8.64mW$ in low-power-run at $26MHs$, while achieves $3.54\mu W$ in deep sleep mode[2]. The processing energy costs for the MCU platforms, as it was done for PULPv3, are computed according to a run-until-completion behavior.

[2] *STM32L476*'s Stop2 Mode is considered as deep sleep mode

Moreover, in Table 3.5 the proposed event-driven system is compared with a traditional frame-based vision system and with an event-based imaging system composed by an *STM32L476* MCU as processing unit coupled with, respectively, an ultra-low-power CMOS imager [87] and an asynchronous retina [12]. Both sensors power consumption are linearly scaled to match the resolution and the frame-rate of the considered imager. For the retina-based system, a time window is defined as long as the frame period and it is assumed to retrieve the same data as the considered imager within this time window. As a consequence of these optimistic assumptions, the system is able to exploit an efficient computational model and can be kept in deep sleep mode for a long time.

For every smart-camera scenario, both the image sensor and the processing platform unit contribute to the system energy cost. **On the low power STM32 MCU, applying an event-driven processing approach results in a** 57.7x **less energy cost with respect to a frame-based scenario, due to the reduction of either processing or imager energy cost**. Thanks to the parallel and efficient operation, the PULPv3 processor is much more energy efficient compared to the others MCUs. Despite the very low-utilization of the available computation power, it reduces the processing energy cost per frame by 14.8x and 5x compared with respectively the STM32 and the Apollo. Figure 3.8 reports the system energy reduction. **By coupling PULPv3 with the imager, the proposed node architecture achieves an energy cost per frame of** 1.79μ**J, providing an overall energy boost of almost** 383x **and** 21.8x**, in terms of energy saving, compared to, respectively, a frame-based visual system and an asynchronous retina-based imaging system**. If powered by a coin cell battery with a capacity of 250mAh at 3V, the proposed smart sensor node will ensure a battery-life of about 248 weeks, thanks to the estimated average power consumption of 17.9μW.

3.6 Summary

This Chapter reported the description of an Ultra-Low Power Smart Camera Architecture composed by an Event-Based image sensor and an energy efficient quad-core processor. The system is benchmarked against a monitoring application to quantify the benefits of exploiting event-based sensing and processing techniques.

Besides the individual low power consumption, the imager features continuous mixed-signal processing to produce significant vision events. When coupled with the Address Event Representation encoding, **the data bandwidth is reduced by** 31x, if compared to an imager that continuously sends full pixel frames.

The event-driven processing of the visual data runs on a fully programmable multi-core platform. This Chapter described the implementation and optimization strategies of the data analytics flow. Compared to most common approaches, based on image processing from traditional frame-based cameras, **the event-driven processing enables a** 57.7x **processing energy-frame saving when running on an ultra-low-power MCU platform**.

When also exploiting the parallel computation provided by the PULP platform, **the overall estimated energy cost of the proposed smart camera system results** 383x **and** 21.8x **lower than, respectively, a traditional frame-based visual system and an asynchronous retina-based imaging system**.

Chapter 4

Smart Camera System-Level Design and Implementation

4.1 Overview

Locally processing data in an energy efficient way is an essential characteristic for smart sensory systems belonging to the IoT ecosystem. Despite featuring power-optimized components, an IoT device must implement an optimal *Power Management Control Strategy* to gain the requested services and performance levels with a minimum number of active components or a minimum load on such components [11]. In fact, when working at the maximum performance level, i.e. in the *Active* state, a component draws a power consumption that can be several orders of magnitude higher than if it is kept in *Idle* or *Sleep* modes. Due to the typical non-uniform behavior of the workload, meaning that various phases alternate at runtime spanning from zero activity to high-workload conditions, it is extremely convenient from an energy perspective to turn a component, or part of it, off if not currently used. Given that, an optimal power management policy aims at minimizing the total energy consumption by controlling the operating modes of the single blocks. Such a power management strategy leads to **keep a Sensor Always-ON while featuring a minimum average power consumption** [4].

From an architectural viewpoint, a system controller coordinates the power management scheme and controls the operating modes of the single components depending on the functional requirements. To be effective, a finely-tuned system power management policy needs to be driven by the computational model of the system. For instance, within a smart sensory system, the sensing and processing units can be put in Active mode during, respectively, the acquisition and the processing tasks, while they can be kept in sleep state for the rest of the time.

A traditional smart camera system relies on a time-driven operating framework, according to which the processing unit performs a frame-by-frame analysis of data periodically sampled by the image sensor [59]. To run an energy-efficient *frame-driven computational model*, the processing unit periodically wake-ups from the deepest sleep state and triggers the image capture of the camera, which is also placed in the idle state before the request. After transferring the image data into the internal memory, the processor runs data analytics and then go back into the sleep mode. The transition from the sleep to the active state is driven by a timer unit featured by the always-on region of the processor architecture. A graphical illustration of the Frame-Driven Computational Model is provided in Figure 4.1a. Although this mainstream approach is effective, it is power-inefficient because it requires transferring image data and wake-up the digital processing unit even if no interesting elements appear on the camera field of view.

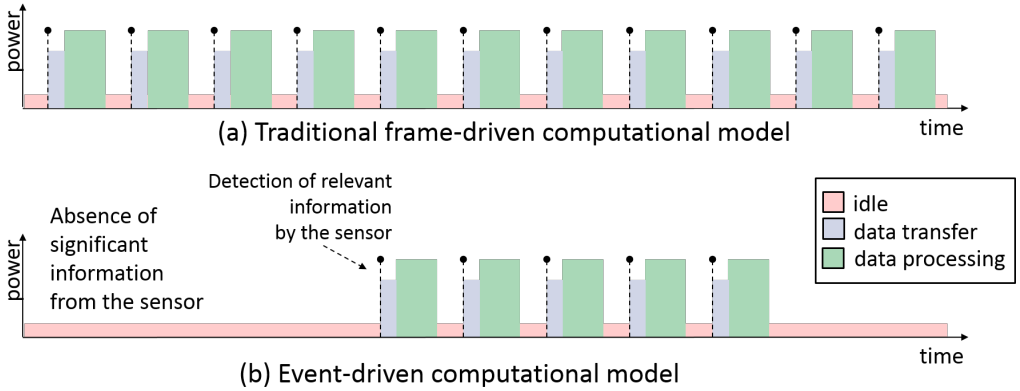


FIGURE 4.1: Power usage comparison between frame-driven and event-driven computational frameworks.

Focal-Plane Processing and Event-Driven paradigms discussed in previous Chapters provide a valuable opportunity to address this issue. Thanks to the extraction of low-level features on the analog side, an image sensor can assess the presence of relevant information at an early stage. Only when detecting a meaningful data, the image sensor itself can drive the wake-up of the digital processor for post-processing actions. In opposition to the frame-driven computational model, this working framework is labeled as *Event-Driven* because the power management scheme is guided by the context-activity. Figure 4.1b schematizes the power usage within a system featuring an event-driven computational model. On average, the power cost results extremely lower than the frame-driven counterpart and, potentially, can reach the idle power in case of rare or very infrequent external events, but still being fully functional.

This Chapter describes the HW architectural solutions to enable an Event-Driven Computational model within the Event-Based Smart Visual Sensor presented in Chapter 3. More in details, the chapter presents:

- The design of a specialized camera interface IP to be integrated within the architecture of the processing unit, which enables a fine-grain control of the operating modes of both the event-based imager and the processing units. Indeed, the power control is moved close to the sensor to favorite a context-aware power management scheme.

- The implementation of the camera interface by exploiting a low-power off-the-shelf FPGA device. Also, a description of the full-system operation is provided to understand the global event-driven power management strategy.

The Smart Visual System system, along with the Event-Driven computational model enabled by the camera interface, is evaluated against three always-on monitoring applications. To this aim, the sensor node operates as a Smart Visual Trigger, hence generating alarms whenever an event-of-interest is detected over the monitored area. The triggering process is also compared with common vision approaches featured by traditional systems with RGB imager sensors. This exploration concludes that the event-driven approach does not lead to performance degradation

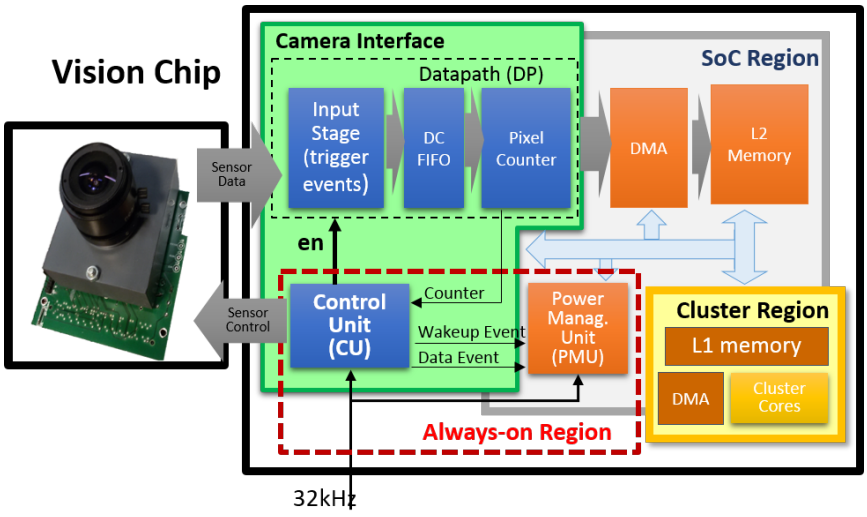


FIGURE 4.2: Block diagram of the Camera Interface IP and its integration within the Processing Unit.

over the considered application case-study unless the scene illumination is considerably reduced. Within the presented monitoring scenarios, the Smart Visual System can reach an overall power consumption as low as $277\mu W$ thanks to the implemented power-management strategies and by selectively activating the digital processing unit. Such consumption is an order of magnitude lower than other reported smart camera systems.

The remainder of this Chapter is organized as follows. Section 4.2 describes the design of the camera interface for enabling an Event-Driven Computational model. The implementation of the camera interface on a low-power FPGA is detailed in Section 4.3 along with the adopted power management strategy. Section 4.4 describes the considered visual applications and the employed triggering process approach. The experimental evaluation, both in term of detection accuracy and power consumption, is reported in Section 4.5 while Section 4.6 provides the summary of the Chapter.

4.2 Design of a Camera Interface for Event-Driven Sensing

This Section describes the design of a Camera Interface IP to be integrated within the Smart Visual System presented in Chapter 3. A block diagram of the peripheral module, which is placed between the vision chip and the processor, is detailed in Figure 4.2. From a system-level point of view, the camera interface is in charge of:

1. Handling the communication between the vision chip and the PULP platform.
2. Autonomously managing the sensing operation and the power modes transitions of the imager without the processor intervention.
3. Driving the system power management strategy and enabling an Event-Driven Computational Model.

As mentioned in Section 3.2.1, the employed Event-Based Ultra-Low Power Imager implements two different operating modes, named *Idle* and *Active*. When operating in *Idle* mode, the sensor outputs a 13 bits counter value related to the number of

events detected during the last integration period (and after the frame difference). Instead, in *Active* mode the sensor outputs a stream of addresses, i.e. the *Events*, which correspond to the coordinates (x,y) of the asserted pixels. In *Idle*, the power consumption is mainly due to the sensor internals and accounts about $10\mu W$ at 10fps and 25% pixel activity. In *Active* mode an additional energy consumption is incurred due to the increased pad activity, which is proportional to the number of asserted pixels and to the frame rate. Given that, the power raises up to $100\mu W$ at 50fps.

The vision chip features a native interface, which consists of control and data IO pins. To operate a continuous focal-plane processing, a periodic control sequence is provided in input, as detailed in [47]. When in *Active* mode, the asserted pixels are dispatched out according to a raster-scan readout process. The triggered events can be sampled by reading the 8-bit Data Bus signal in correspondence of the rising edge of the Write-Enable output pin. This signal features 7 bits representing the column coordinate (y-coordinate) and 1 bit for the sign. An additional End-Of-Row (*EOR*) output pin indicates the increase of the row index during the readout process. Therefore, the x-coordinate of an event is computed by accounting the number of EOR pulses up to the pixel reception. The Event stream is considered terminated when the *EOR* signal have pulsed for 64 times (the number of rows). Note that the peak data rate reaches $80Mpixel/sec$ during the readout process, leading to a complete readout time of less than $300\mu sec$.

To connect with the Event-Based Camera, the proposed camera interface, which is integrated within the PULP architecture, contains a Control Unit (CU) and a DataPath (DP) subsystems. The CU drives the sensor control signals depending on the current readout mode and the programmable internal register values (e.g. the integration time value). Moreover, the CU manages the transition scheme between *Idle* and *Active* modes based on a user-defined threshold, as also proposed in the work of Gasparini et al. [43]. More in details, when the sensor is in *Idle* mode, the camera interface readouts the event counter through the sensor data bus. If the value overcomes the specified threshold, the CU switches the imager to the Active mode by generating the proper timing signals. To enable an always-on sensing, the CU must be permanently active and driven by a low-speed free-running clock. To this aim, the Control Unit is placed within the always-on region of the SoC region, which is fed by the external 32kHz clock.

On the other side, the Datapath (DP) serves for (a) triggering data coming from the sensor when in Active mode and (b) transferring them to the L2 memory of the processing unit. Such a DP firstly converts visual data from the native protocol to a *(x,y)* format, through the *Input Stage*, and then counts them (through the *Events Counter*). An additional *Dual-Clock FIFO* (*DC-FIFO*) is needed to cross the boundary between the imager clock domain and the internal Data Path clock domain. The DP block features opposite requirement in terms of clock speed with respect to the CU since it needs to be sufficiently high to sustain the sensor peak output rate of $80Mpixel/sec$ during the short readout period. For this reason, the Data Path clock is driven by the SoC clock.

When the sensor is in *Idle* mode, the SoC region, including the Datapath of the Interface but not the always-on region, and the Cluster are kept in deep sleep mode. If the CU, that is continuously running, detects an increasing amount of events, i.e. a number of events above the threshold, it switches the imager to the *Active* mode to enable the output dispatch. Before the readout phase, the CU also issues a *Wake-Up Event* to the PULP Power Management Unit (PMU) to request the SoC activation, and therefore the activation of the camera interface DataPath. Once the readout process is completed, the DP is disabled and all the received data are available in the

TABLE 4.1: FPGA Resource Utilization

Resource	Used	Total	Percentage
VersaTiles [1]	3758	6144	61.17%
RAM 4,608-Bit Blocks	8	8	100%
VersaNet Globals [2]	6	18	33.33%

L2 memory. At this time, the *Data Event* can be asserted to trigger data analytics on the Cluster.

By means of the proposed system architecture and the camera interface, the Event-Driven computational model can be enabled. Indeed, the system is fully activated only when a relevant information is detected by the sensor (a number of events above a predefined threshold). During the rest of the time, instead, the components are kept in a Idle state to consume a minimal power. Still, the continuous sensing operation is guaranteed thanks to the imager and the camera interface.

4.3 System Implementation and Power Management

Camera Interface FPGA Implementation. To enable the communication between the prototyped PULPv3 chip and the Event-Based Imager, the camera interface architecture has been adapted to be implemented on a low-power flash-based FPGA Microsemi's IGLOOnano AGLN250V2 [81]. Differently from the architecture presented in the previous chapter, the Camera Interface contains an SPI Slave interface through which the processing unit can set up the acquisition parameters (e.g. the frame-rate and the exposure time) and read the data. Moreover, a *Storage First-In-First-Out (FIFO) Buffer* is included into the DataPath to temporarily store the data before the SPI readout. As discussed before, two different clock domains are defined for the CU and DP subsystems. The CU is still fed by low-speed free-running 32 kHz clock to allow a continuous sensing. Instead the DP clock, which needs to be sufficiently high to register the events coming from the image sensor, is internally generated by means of a ring-oscillator. To save power, this high-speed clock can be enabled only for the short readout period, which lasts less then 300μsec per frame, if the sensor is in Active mode. Considering a frame period of 100msec (corresponding to 10fps), a power saving of 39x is achieved by disabling the ring-oscillator for the majority of the frame time with respect to keep both the clock drivers continuously active. When the FPGA core works at 1.2V, which is the lowest nominal supply voltage, critical timing issues are reported if the high-speed clock is at 80Mhz, which corresponds to the peak sensor output rate. To overcome this bottleneck, the DP *Input Stage* features a shift register to gather 4 incoming events, which are then pushed into the DC FIFO. Thus, to relax the timing constraints, the ring-oscillator frequency is set to 25MHz, which results to be sufficiently high to handle the transfer of 4-events packets to the Storage Memory. Due to the FPGA resource constraints, the internal memory buffer (Storage Memory) has been sized to collect a maximum amount of 1024 events, corresponding to 12.5% of the theoretical maximum amount, which is sufficient to store the amount of pixels in most common typical monitoring application scenarios, as shown in the previous chapter. Tab. 4.1 reports the FPGA resource utilization.

[1] Equivalent to a three-input lookup table (LUT) or a D-flip-flop/latch with enable
[2] Global clock distribution network

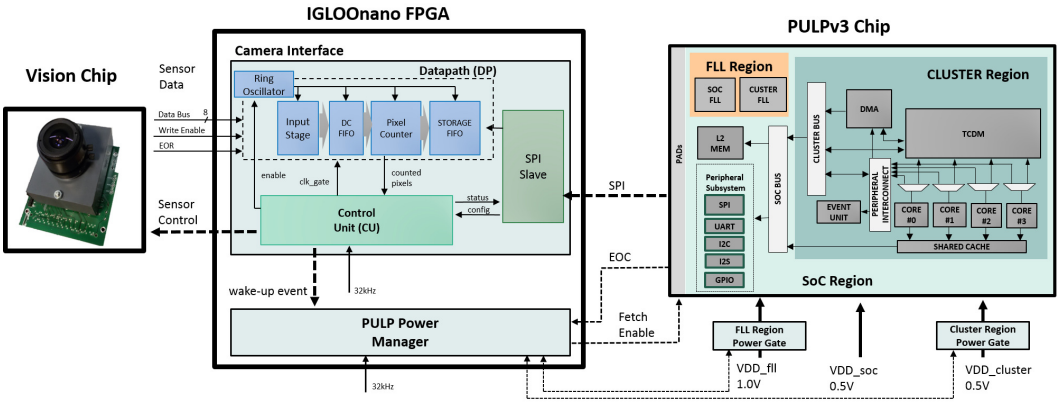


FIGURE 4.3: System Architecture and Block Diagram of the Camera Interface when implemented as external component with a Low Power FPGA.

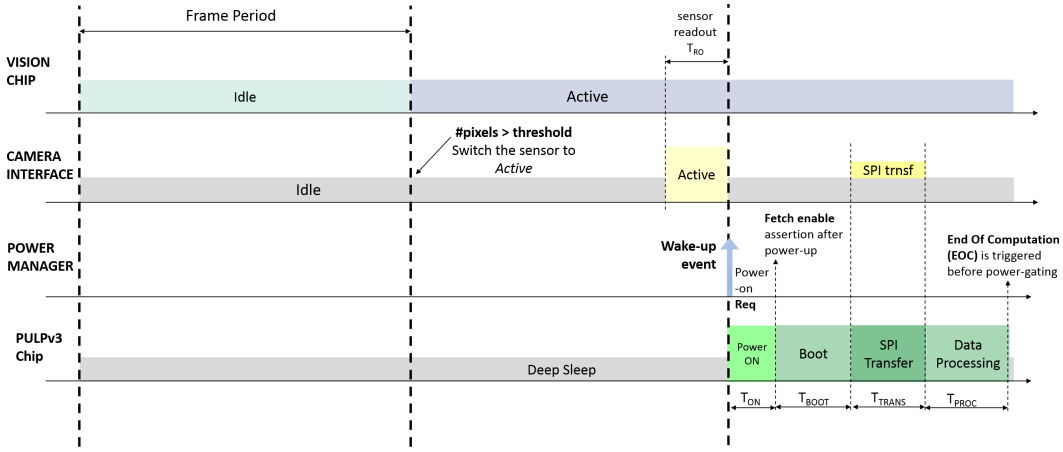


FIGURE 4.4: Power management strategy within the smart camera system.

TABLE 4.2: Power consumption of the system components

Component		Fully-Active Power	Idle Power	Note
Vision Chip		$20\mu W$	$10\mu W$	Measured at 10fps and 25% activity [47]
FPGA Camera IF		$3mW$	$68\mu W$	Additional $456\mu W$ due to SPI transfer (5MHz)
PULPv3 Chip	Cluster Region	$946\mu W$	-	At 0.5V, 30MHz
	SoC Region	$313\mu W$	$99\mu W$	At 0.5V, 30MHz
	FLL Region	$3.2mW$	-	At 1V

System Power Management. The vision chip, once coupled to the camera interface, enables a context-aware system-level power management strategy. According to this Event-Driven approach, the device exploits information from the context to select the appropriate power mode. This concept is is graphically illustrated in Figure 4.4 when applied to the considered system. When the sensor is in *Idle* mode, the other components are put in the lowest power saving mode. In this state, the system still continuously senses the environment thanks to the vision chip and the camera interface operation. Differently from common frame-based vision systems, the digital processor is normally kept in deep-sleep mode and activated depending on the context-activity. When the amount of sensed data overcomes the pre-defined threshold, the camera interface switches the sensor to *Active* mode and then triggers the processor wake-up. Within the interface, the ring-oscillator is only activated within the readout phase, leading to a small increment of the average power consumption with respect to the Idle power state. After completing the readout, the camera interface issues a *wake-up event* to the external power manager. This latter requests the power-on of the platform, by triggering an activation signal to both the power-gates of the cluster and the FLL regions.

Tab. 4.2 reports the measured components power consumption, either in Fully-Active and in Idle states. When operating at the peak capacity (with DP clock running), the camera interface consumption is dominated by the dynamic power associated to the high-speed clock domain. When the ring oscillator is disabled, the FPGA power is due to the leakage and the 32kHz clock activity, resulting in an extremely lower value of $68\mu W$. Therefore, to save power when the sensor is in *Active* state, the ring-oscillator is activated only during the short period of data readout. Acting such a duty cycle technique, an average power of $77\mu W$ is achieved within the frame period. In addition to this, a power contribution of $456\mu W$ is accounted during the SPI data transfer. In the Idle state, instead, the FPGA power consumption is limited to $68\mu W$.

Within the *PULPv3* processor chip, both the Cluster and the SoC regions are powered at 0.5V while the FLL region requires a higher supply voltage of 1V. The Idle state of the processor refers to the implemented deep-sleep mode. Within such state, the FLL and the Cluster regions are power-gated, while the SoC region remains continuously powered to guarantee data and executable code retention on the L2 memory. This is essential for a fast start-up process since loading the executable (45kBytes) from an external non-volatile memory requires 1.7msec through a quad-SPI interface at 50 MHz [1]. Moreover, even assessing the increased latency as a non-critical issue, the energy consumption associated to the read transfer operation from the external flash memory results to be 7x higher than the energy consumed within the frame period due to the leakage of the always-on SoC region.

TABLE 4.3: System Delay Timing Parameters

Parameter	Description	Value [μsec]
T_{RO}	Sensor readout Time	300
T_{ON}	Power-on and FLL lock time	590
T_{BOOT}	Boot-up process time [30 MHz]	61

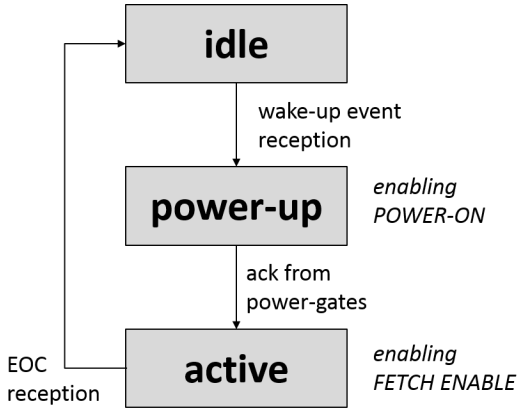


FIGURE 4.5: PULP External Power Manager state machine

A *Power Manager* controller is implemented on the FPGA, along with the camera interface, to manage the transitions between the Active and Deep Sleep modes of the PULP processor, through discrete onboard power gate components. The internal state machine of the PULP power manager is illustrated in Figure 4.5. After the reception of the *wake-up event* from the camera interface, the power manager is in charge of powering-on the PULP processor by controlling the power-gates. The fetch-enable signal is then asserted after the reception of the acknowledge from the power-gates components. Eventually, the controller goes back in *Idle* state once the *End-Of-Computation* (*EOC*) signal is asserted by the PULP processor. Table 4.3 lists the start-up timing delays of the PULP platform when powered-on. Specifically, T_{ON} refers to the power-on time of the supply voltages and the FLL lock time when in the closed-loop configuration. T_{BOOT} indicates the boot-up time of the processing platform. These start-up phases determine a fixed energy cost for every activation of the digital processor

4.4 Smart Visual Triggering Applications

To evaluate the proposed design, the event-based smart camera acts as a Smart Visual Trigger, aimed at generating alarms when an event of interest is detected in the camera field of view. Depending on the user preference, a control action can take place after the alert generation, such as a counting operation or the activation of a secondary high-resolution RGB camera.

The trigger generation process runs on the digital processor upon the reception of a wake-up signal from the imager, therefore only if a relevant motion is early detected within the imaging pipeline. From a high-level viewpoint, the visual processing flow is composed of the following steps.

1. *Object Detection*. The event-driven process presented in Section 3.3 is exploited to detect moving objects within the camera plane. As previously highlighted,

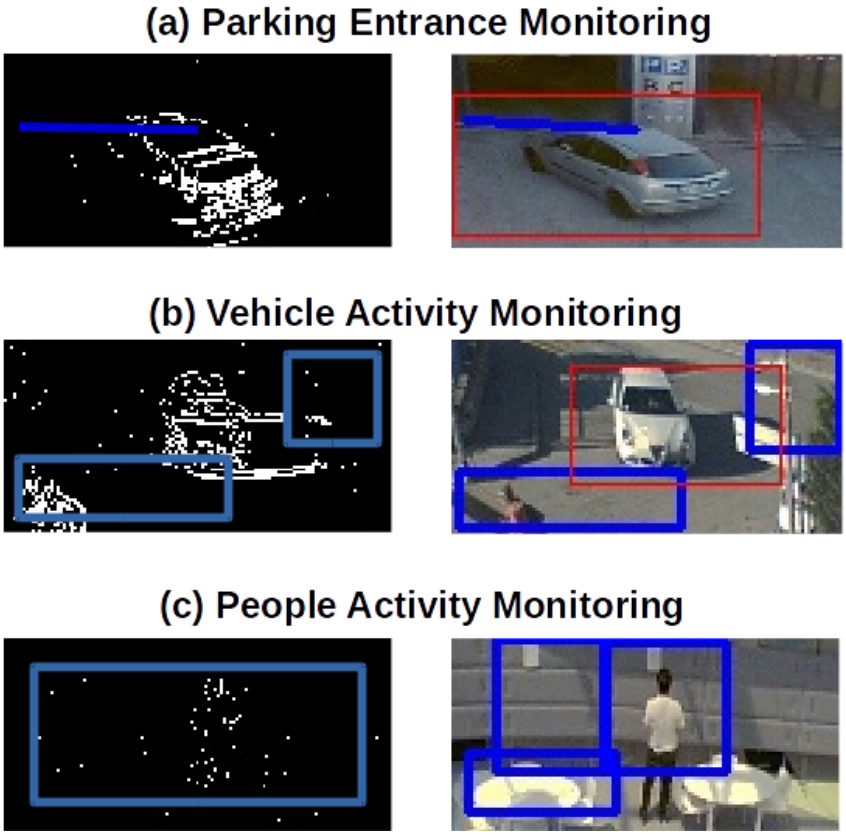


FIGURE 4.6: Monitoring scenarios for the considered applications. Frames on the left are captured by the event-based imager (active pixels are drawn in white) and frames on the right come from a commercial RGB camera.

this task analyzes the stream of AER *(x,y)* events triggered by the Event-Based ultra-low power camera.

2. *Object Tracking*. A Kalman filtering scheme is implemented on top of the event-driven processing to keep track of the detected moving objects along with their temporal properties. To this aim, every tracked object is modeled with a Kalman Filter. The assignment problem between any moving object and the existing Kalman models is solved through the Hungarian algorithm [62] after defining a cost function $cost(i,j)$ between the *i-th* object and the *j-th* tracker as $cost_{i,j} = \lfloor d_{L_1}(C_i, C_j) \rfloor + d_{L_1}(\phi_{max,i} - \phi_{min,i}, \phi_{max,j} - \phi_{min,j})$. An assignment is performed only if the cost value is lower than a threshold. The resolution of the optimization problem is handled by a sequential task, while the update process of the Kalman filters is split over the available cores.

3. *Triggering*. An alert signal is generated whenever one of the tracked objects matches a predefined condition. For instance, if the center of mass of a tracked objects crosses a virtual gate or enters a virtual loop in the camera plane.

To assess the performance of the presented smart visual trigger, three environmental monitoring scenarios are considered and listed below. Figure 4.6 illustrates some frames captured from each scenario.

- **Parking Entrance Monitoring.** The outdoor entrance of a parking space is monitored to detect cars entering from the left gate. The triggering process is

TABLE 4.4: Monitoring Application Dataset Statistics

Application	#frames	#events
Parking Entrance Monitoring	4000	9
Vehicles Street Monitoring	4000	63
People Activity Monitoring	1446	19

activated when a moving object crosses the virtual gate corresponding to the parking target entrance (Fig. 4.6a).

- **Vehicle Street Monitoring.** The smart trigger is employed to monitor vehicles at a crossroad. Here, the triggering condition is associated with any tracked vehicle entering one of the defined virtual loops corresponding to different crossroad directions (Fig. 4.6b). In this scenario, the presence of shadows due to the high illumination makes trigger generation a more tricky task, possibly causing extra alert signal assertions.

- **People Activity Monitoring.** The third application deals with people crossing an indoor area and possibly stopping by a landmark or a point of interest (Fig. 4.6c). To detect these latter events, an alarm is triggered when any detected and tracked object disappears from the tracking list. Motionless objects are not detected by the sensor because of the internal frame difference operation. Therefore, any tracker associated with moving people is expected to disappear when a person stops. To increase the reliability of the trigger generation, trackers disappearing at the frame borders do not cause any alarm since they are most likely associated with people exiting the scene. Likewise, limited spatial movements of tracker's center of mass, such as those generated by people standing still, is filtered to avoid generation of false alarms.

4.5 Evaluation

Smart Trigger Generation. The trigger generation process has been evaluated on a dataset of images taken for all the considered applications. The frame rate of the camera is set to 10 frames per second. The evaluation has been initially conducted with nominal environmental conditions (i.e. good lighting). For every scenario, the camera has been placed in a fixed position to not degrade the motion estimation, which is internally performed on the sensor side through the frame difference operation. During the acquisition sessions, a secondary RGB camera has been used for the ground truth (Sony Playstation Eye™ camera).

Table 4.4 reports the characteristics of the visual dataset for the three monitoring scenarios, in terms of the number of frames and occurring events of interest. Every raised alert signal is manually marked as True Detection (*TD*), when an event of interest is correctly detected, False positive (*FP*), if an unwanted trigger is generated, and False Negative (*FN*), in case of miss-detection. Based on that, the following metrics are computed for the evaluation:

$$precision = \frac{\#TD}{\#TD + \#FP} \tag{4.1}$$

$$recall = \frac{\#TD}{\#TD + \#FN} \tag{4.2}$$

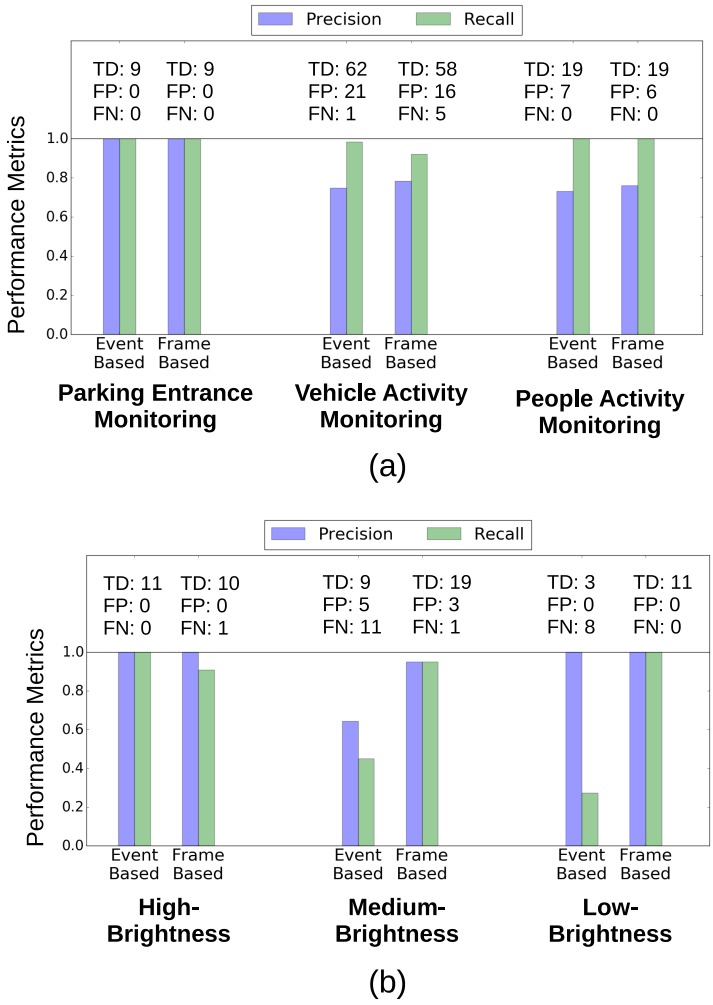


FIGURE 4.7: Performance metrics for event-based and frame-based implementations on dataset from (a) monitoring applications and (b) on different lighting exposures

where # refers to the events' cardinality. Note that both the indicators should be equal to 1 for an optimal detection.

For comparison purpose, a basic triggering process is implemented on the ground truth RGB images, following a typical computer vision flow such as [22]. Every RGB frame is transformed and scaled to match the resolution of the Event-Based imager. Pixel values are also converted into a grey-scale representation. For any of the considered applications, objects are extracted by computing the connected components after applying a background subtraction, based on a Mixture of Gaussians (for vehicle detection and street monitoring) or frame difference (for people activity monitoring), and morphological filters. Blobs with a low number of pixels are filtered out to improve detection accuracy. Then, an alert signal is triggered whenever a tracked object matches a predefined condition. Tracking is obtained by associating objects from successive frames based on the centroid's distance and the bounding box's size differences.

Figure 4.7a reports the statistics and the performance metrics computed for the triggering process on the two domains: event-based and frame-based. The *Parking Entrance Monitoring* application shows optimal precision and recall on both the

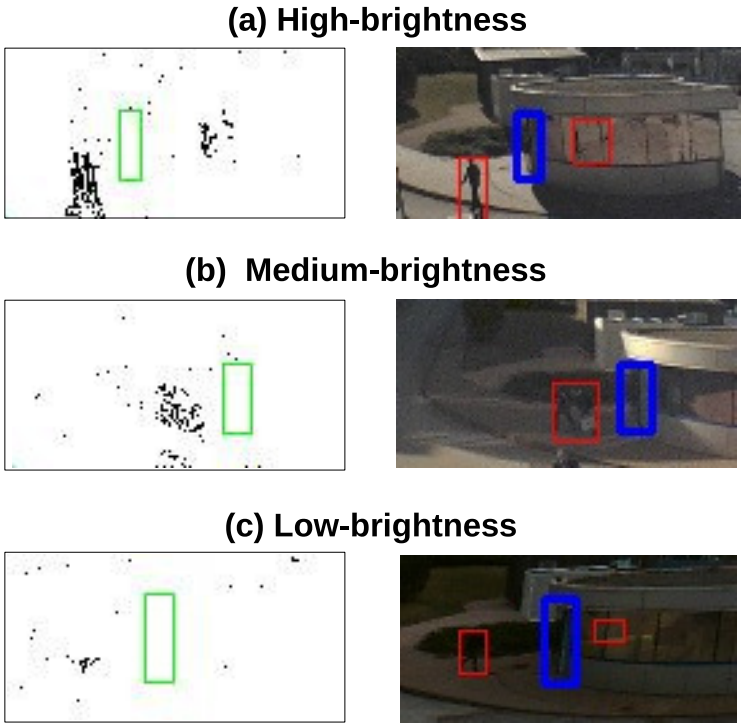


FIGURE 4.8: Monitoring scenarios with different light exposure. Frames on the left are captured by the event-based imager (active pixels are drawn in black) and frames on the right come from a commercial RGB camera.

domains because of the clear vehicle appearances. On the contrary, the scenario addressed for *Vehicle Street Monitoring* is more challenging. Both the event- and frame-based implementations suffer from a high number of false positive alarms caused by moving people in the scene. Along with it, the event-based approach also presents extra *FP* alarms due to the fact that a single vehicle can be tracked by multiple blobs, each one attached to different parts of a vehicle. On the other side, the frame-based approach shows a higher precision but lower recall. The increased number of false negatives arises from the loss of trackers due to the merge of close vehicles after background subtraction and morphological filtering. The *People Activity Monitoring* application presents similar performance metrics for both domain implementations. False Positives are caused mostly by slight movements of people standing still or crossing persons. Based on these results, the performance triggering accuracy of the Event-Based system is not degraded with respect to a typical frame-based approach within the evaluated environmental scenarios.

To evaluate the robustness of the proposed system to different light exposures, the *People Activity Monitoring* application is tested against an additional video dataset, collected in an outdoor scenario. The considered application involves the monitoring of people entering and exiting a building gate. A trigger signal is generated whenever a person or a group of compact people enter or exit the virtual loop corresponding to the door. Figure 4.8 shows the targeted scenario with three different light conditions: high-brightness (captured at midday), medium-brightness (early in the morning) and low-brightness (late in the evening). Figure 4.7b reports the statistics of the triggering process within the above-mentioned scenarios, along with a comparison with the same approach in the frame-based domain. In this case if low-light conditions, performance metrics appears degraded with respect to the

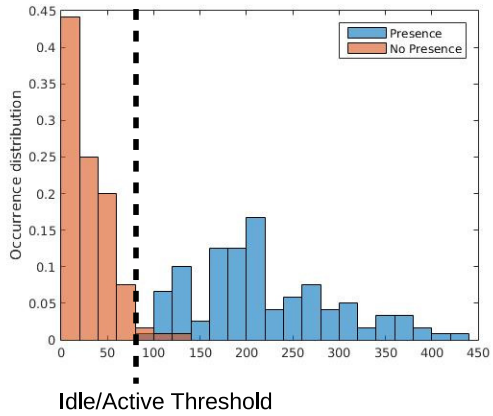


FIGURE 4.9: Tuning of the wake-up pixel threshold by observing the distribution of the number of pixel for frames that contains objects (blue class) and not (red class). The class distribution is built by picking 120 uncorrelated samples per class from the "People Activity Monitoring" dataset.

frame-based approach. This is a consequence of a reduced spatial contrast within the images. Most likely, the object detection processing of Section 4.4 fails because of the low number of input data. On the contrary, background subtraction of the frame-based approach still performs properly, independently from the environmental lighting.

Power Consumption. In the following, the evaluation of the proposed event-based system is reported in terms of power consumption. To this purpose, measurements have been taken from a system prototype in correspondence of the power modes described in Section 4.3. When fully-active, the smart camera system presents a power consumption of 7.62mW, mainly due to the contribution of the PULP platform and the FPGA interface. Individual power measurement values are indicated in Tab. 3.5. Thanks to the lightweight application signal processing, the processor is not required to run at the maximum speed all the time. Therefore, as a first optimization step, the digital processor is activated after every frame acquisition and then put in the deep-sleep mode when the data processing action completes. Such kind of computational framework is here referred as *Periodic-Polling*. In addition to this, the high-speed clock of the FPGA camera interface is enabled only for a short readout period. By applying this power management strategy, the average power consumption of the individual components is extremely reduced. The FPGA camera interface achieves an average power of $77\mu W$, mainly dominated by the consumption associated with the 32kHz clock domain. The processor power in sleep mode is due to the SoC region leakage power of $99\mu W$. In active mode instead, the energy consumption depends on the processing time, the data transfer operation and the start-up process, which accounts the power-up, the FLL activation and the boot process. For frames composed of a minimal amount of events, these latter contributions dominate the whole energy cost. Table 4.5 contains the average power consumptions values on the monitoring application when applying the described power management strategy. With respect to a fully active system, a power reduction of at least 25x is achieved thanks to the periodic-polling framework.

The second optimization step concerns shifting the computational model from the *Periodic-Polling* to the *Event-Driven* framework, which is enabled by the designed

TABLE 4.5: Power Consumption within the considered Applications Case-Studies

Application	Event Threshold	% relevant frames	App Duty Cycle	Periodic-Polling Power	Event-Driven Power	Reduc.
Parking Entrance Monitoring	100	16%	0.7%	226μW	193μW	-14.6%
Street Traffic Monitoring	40	60.5%	1.1%	294μW	277μW	-6%
People activity Monitoring	80	65.4%	1%	267μW	252μW	-5.6%

camera interface. The Event-Driven computational model is intended to keep the processor in deep sleep mode until the sensor produces a relevant amount of motion data from the external context. Consequently, the camera interface transfers data to the processor unit only if the frame contains at least a specific number of meaningful data. The wake-up threshold value corresponds to the value defined within the filtering stage of the blob detection module. Basically, the system is not awake when any objects will not surely be detected by the post-processing operation because of the low-amount of produced spatial-contrast pixels. The mentioned filtering parameter is manually tuned based on the average amount of pixels describing the objects of interest. Within the *People Activity Monitoring* application, a further analysis is done to investigate a finer tuning. Figure 4.9 shows the probability distribution of the number of pixel in frames belonging to two classes: the red one includes frame with no object presence, while the blue one corresponds to frames with the presence of one or more moving object. Based on that, the threshold is set slightly higher than the filtering value (80 instead of 40), leading to higher power savings because of the reduced number of activations of the digital processing sub-system.

When motion is not detected in the camera field of view, the system exploits the Idle state as described in Section 4.3. In this state, the vision chip works in Idle and the ring-oscillator of the FPGA camera interface is not enabled. Considering also the processor leakage due to the SoC region consumption, an overall Idle power of 176.88μW is achieved. Potentially, the average power consumption of the system converges to this bottom limit when no meaningful information is detected within the context. Tab. 4.5 highlights the power reduction considering a selective processor activation in correspondence only to the subset of relevant frames (those with a number of events higher than the defined threshold). Clearly, a higher reduction is assessed for the application featuring a lower percentage of interesting frames (16%). In this scenario, a power consumption of 193μW is achieved, closer to the Idle power lower bound. Instead, within the other application scenarios, a power consumption between 252μW and 277μW is reported, coming from a percentage of processor activation slightly higher than 60%. In these cases, a power saving of about 6% is achieved if compared with the correspondent periodic-polling computational framework.

Figure 4.10 depicts the average power consumption breakdown on the two considered computational frameworks. The higher energy-efficiency of the event-driven computational model mostly arises from the power saving exploited within the Cluster region and the FLL region. More precisely, a power saving of about 65% is accounted within the first application in the two mentioned power domain regions. Instead, a 15% reduction is reported for the other two applications, because of the higher processor activation rate. The saved energy is mostly related to the energy fixed costs regarding the processor wake-up procedure (power-up, FLL activation and boot). In addition to this, an FPGA power saving between 5% and 10% is accounted, due to less high-speed clock activations and SPI transactions.

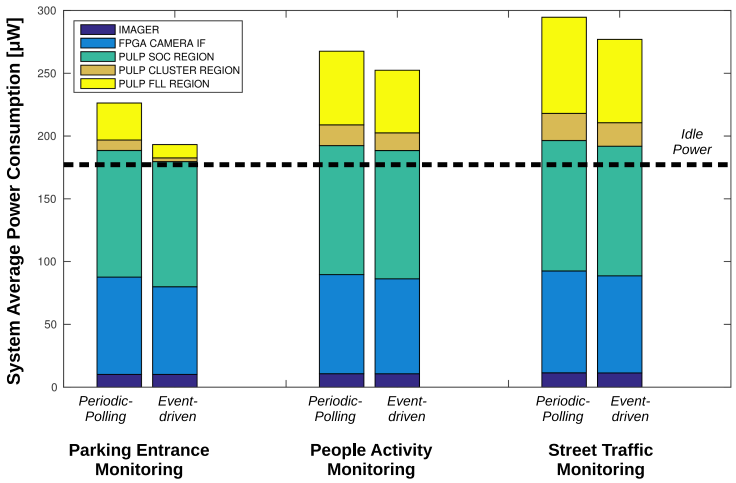


FIGURE 4.10: Power consumption breakdown of the system.

Comparison with Related Works. Many of the existing smart camera nodes feature a high computational power to deal with the complexity of computer vision algorithms [25, 17, 88, 56, 13, 33]. These architectures include power-hungry components to handle image acquisition and frame-by-frame data analytics. For this reason, the overall power consumption reaches several hundreds of mW, which is much higher than the power cost of the event-based visual system presented before.

The achieved sub-mW average consumption is enabled by leveraging novel techniques, such as focal-plane processing, and the event-driven local computation. In contrast, the system Wi-FLIP [40], which also includes a mixed-signal imager, presents a not power-optimized processing unit and draws a power of more than 100mW. Other systems present an event-driven model to lower the energy consumption. The camera presented by Magno et al. [77] triggers the image acquisition upon the detection of human movements by using a PIR sensor, which however carries a much lower information and consumes a higher power than the event-based camera when in *Idle* mode. MeshEye [51] proposed a low-resolution stereo camera for triggering scope, but it is still drawing more power than the proposed solution.

Close to the Event-Based smart visual camera discussed in this work are the nodes proposed by Gasparini et al. [43] and Carey et al [16]. The first one couples the same event-based imager with an IGLOO FPGA, which implements either the camera interface and the digital processing circuit. This design features a dual-clock domain, where the high-speed clock, that drives the data transfer and processing circuits, is clock-gated to save power if the imager is in the Idle state. By doing this, the power consumption can be reduced from 4.2mW to 2.5mW. However, the system lacks flexibility because of the hard-wired event-driven processing, specialized in counting single persons passing in front of the camera. The system proposed in this work, in contrast, allows to detect and track multiple moving objects and it is extremely flexible thanks to the software fully-programmable architecture. The smart camera presented by Carey et al. features a mixed-signal imager with SIMD programmable processing capabilities coupled together with an FPGA interface and an MCU processing unit. The authors proposed a power-optimized frame-driven computation, leading to a power cost of 5mW within a monitoring application. With respect to this, the presented event-based camera shows an average power consumption 10x lower, thanks to the optimized system power management and the event-driven computation.

4.6 Summary

This Chapter discussed the system integration steps to enable an optimized Event-Driven Computational model. To this aim:

- A specialized camera interface is designed to handle the sensing process and to drive the power management policy of the system. Thanks to this, the most power-hungry blocks of the visual architecture can be kept in sleep mode unless a relevant information (motion) is detected on the camera field of view. Despite the minimal activity of the system when working at the lowest power mode, the system is still actively sensing the environment.

- To deploy a power-optimized smart camera prototype, the camera interface is implemented on a low-power flash-based FPGA. When tested as a smart visual trigger, the presented system show a power consumption of up to $277\mu W$, which results to be more than 10x lower of other state-of-the-art devices. This is possible thanks to the Event-Driven computational model, enabled by the camera interface, and the Event-Based ultra-low power imager.

Chapter 5

Event-Based Binarized Neural Networks

5.1 Overview

The design process of battery-powered IoT devices must trade-off between performance and energy consumption. Shifting from a traditional frame-based to a neuro-inspired event-based paradigm provides a big opportunity to reduce the energy cost of visual sensing. As discussed in previous chapters, the average power consumption can be brought below the mW level by means of event-based sensing and processing and a finely optimized system-level design integration. Thanks to this strategy, a visual system with triggering capabilities has been introduced to be part of the IoT stack. This Chapter further investigates the design the data processing tasks to enhance the visual capabilities of the node and refine the triggering process. The challenge consists of pushing an additional intelligent layer on top of the optimized event-driven computation without degrading the achieved energy-efficiency level.

In the field of Computer Vision, Deep Learning techniques are nowadays the leading technologies for enabling artificial vision [65]. Bio-inspired *Convolutional Neural Networks* (*CNNs*) achieve state-of-the-art performance on several recognition tasks, such as image classification, object detection and voice recognition. Despite the promising capabilities, the high computational cost and the high memory requirements are preventing their deployment on resource-constrained and ultra-low power devices.

To address this issue, recent approaches have investigated the quantization of CNN models to reduce the storage and computational costs of the inference engines. As an extreme approximation, *Binarized Neural Networks* (*BNNs*) makes use of a 1-bit precision to represents both the weights and the values of the activation layers of the network [29, 93]. This implicitly reduces the memory requirement by 32x with respect to full precision CNNs, while not significantly degrading the classification accuracy on several public datasets [29]. By reducing the internal precision to a single bit, the convolution operation, which is the most demanding kernel of a CNN inference algorithm, transforms into a bit-wise logic XNOR followed by a bit counting operation. This has enabled, in addition to the less constrained memory requirement, the deployment of deep networks into low-power programmable processors, also thanks to their flexibility and easy programming legacy [78]. However, due to the stringent energy requirement of always-on sensors, a BNN inference engine needs to be be coupled with power-optimized smart sensing devices to further reduce the system power consumption.

This Chapter provides **the specification of a system that couples the visual performance of Binarized Neural Networks with the energy gains enabled by Event-Based sensing and processing models.** The approach is referred to as *Event-Based*

Binarized Neural Network. By leveraging a mixed-signal imager, the early stages of the convolution operations can be moved into the analog domain of the image sensor. This reduces the workload of the digital processor along with the overall energy cost. With respect to other mixed-signal designs [66, 73], the proposed approach features the in-sensor computation of binary convolutions at an ultra-low power cost. Indeed, concerning the employed Event-Based binary imager [47], it is argued that the performed spatial contrast and binarization of Figure 3.2.1 reflect the operations of a binarized pixel-wise convolution and can be seen as embedding the first binary convolutional layer within the image sensor die.

In the follows, a mixed-signal HW-SW codesign of an *Event-Based Binarized Neural Network* is described and implemented on the smart camera architecture described in Section 3. To this aim, a BNN is trained and tested on binary data produced by the Event-Based binary spatial constrast imager. A software implementation of the BNN is optimized to run efficiently on the programmable 4-core RISC processor. To assess the performance of the proposed Event-Based Binarized Neural Network solution, a real-world dataset collects image samples belonging to three different classes. During the classification task, the proposed solution reduces the system energy by 17.8% in reference to a frame-based baseline system that includes a low-power RGB imager and a traditional BNN approach, while paying only a 3% reduction of classification performance on a 3-classes scenario. Moreover, when considering a long-term monitoring application, the system can leverage the event-based sensing scheme to reduce the start-up activity of the processor and to trigger a classification run upon the detection of a relevant event. This leads to an energy reduction of up to 8x with respect to the frame-based system featuring an RGB camera.

As a summary, this Chapter describes:

1. The specification of an Event-Based Binarized Neural Network model, which fits the energy requirements and resource constraints of deeply-embedded always-on visual sensing front-end.
2. An optimized implementation of a BNN on a 4-core embedded processor.
3. The energy evaluation of the proposed solution and the comparison with a baseline BNN model with RGB data input.

The remainder of the Chapter is organized as follows. Section 5.2 provides some background of Binary Neural Networks. Section 5.3 reports the insight of the proposed Event-Based Binarized Neural Network, while an optimized implementation is detailed in Section 5.4. Section 5.5 reports the experimal results and lastly the Section 5.6 summarizes the chapter.

5.2 Binarized Neural Networks

A *Convolutional Neural Network* model is structured as a sequence of convolutional and fully-connected layers, where the convolution operation demands the highest workload of the inference task. This operation produces OF output feature maps o by applying a non-linear function $f(\cdot)$ to the pre-activation values φ, obtained by convolving the IF input feature maps I with a battery of weight filters w, as expressed by Equations 5.1 and 5.2.

$$\varphi(x,y) = \sum_{d,i,j} w(d,i,j) * I(d,i,j,x,y) \tag{5.1}$$

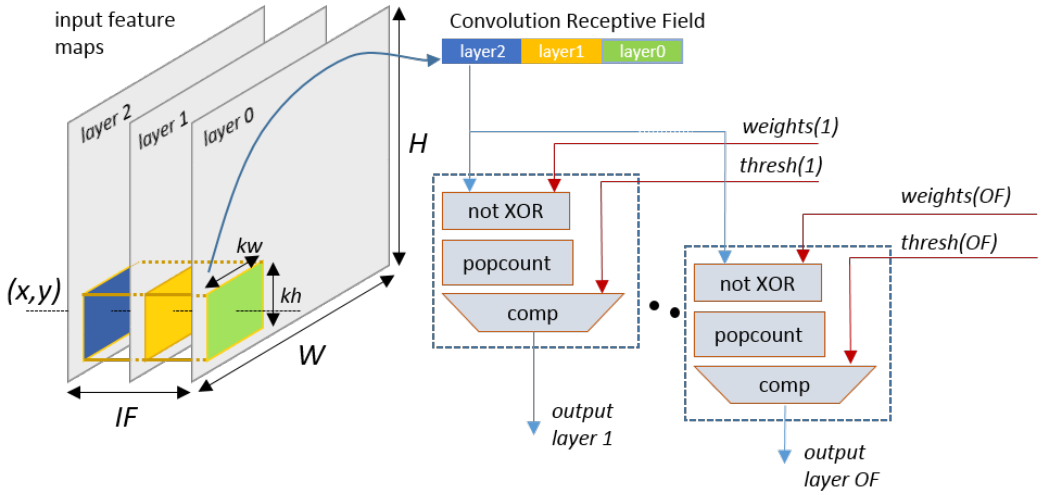


FIGURE 5.1: Binary convolution flow for every convolutional layer. For any of the OF output feature maps, the binary value at position (x,y) is produced by overlapping the mth weight filter to the array of the receptive field of the input feature map centered at the spatial position (x,y).

$$o(x,y) = f(\varphi(x,y)) \tag{5.2}$$

A *Binarized Neural Network* aims at binarizing the synaptic weights w and neuron values o by opportunely training a convolutional network. When looking at the inference task, a binary convolution operation is expressed as Equation 5.1 where $I, w \in \{0,1\}$. Due to this constraint, the operation can be rewritten as:

$$\varphi(x,y) = \text{popcount}(\mathbf{w} \text{ xnor } \mathbf{I}) \tag{5.3}$$

where $\mathbf{w}, \mathbf{I}$ are binary arrays that store the binary filter weights and inputs and popcount$(\cdot)$ returns the numbers of asserted bits of the argument. Note that the convolution output $\varphi(x,y)$ is an integer value.

As presented by the original paper [29], the pre-activation value is binarized after a batch normalization layer, by means of the $sign(\cdot)$ function.

$$o_{bin}(x,y) = \text{sign}\left(\frac{\varphi(x,y) + b - \mu}{\sigma} \cdot \gamma + \beta\right) \tag{5.4}$$

where b is the convolution bias and μ, γ, σ and β are parameters learned by the batch normalization layer. These parameters are floating point but thanks to the $sign(\cdot)$ and the integer input $\varphi(x,y)$ the expression can be reduced to:

$$o_{bin}(x,y) = \begin{cases} \varphi(x,y) \geq thresh(m) & \text{if } \gamma > 0 \\ \varphi(x,y) \leq thresh(m) & \text{if } \gamma < 0 \end{cases} \tag{5.5}$$

where $thresh(m)$ is computed offline after the training procedure as $\lfloor \mu - b - \beta \cdot \sigma / \gamma \rfloor$ if $\gamma > 0$ or $\lceil \mu - b - \beta \cdot \sigma / \gamma \rceil$ if $\gamma < 0$. Therefore only an integer threshold and a single bit sign(γ) have to be stored for binarizing any output layer.

As shown by equations 5.3 and 5.5, a binarized convolution operation is then expressed as a couple of instructions, requiring bit-wise and bit counting (popcount) operations, along with an integer comparison. A binarized convolutional layer computation flow is illustrated in Figure 5.1.

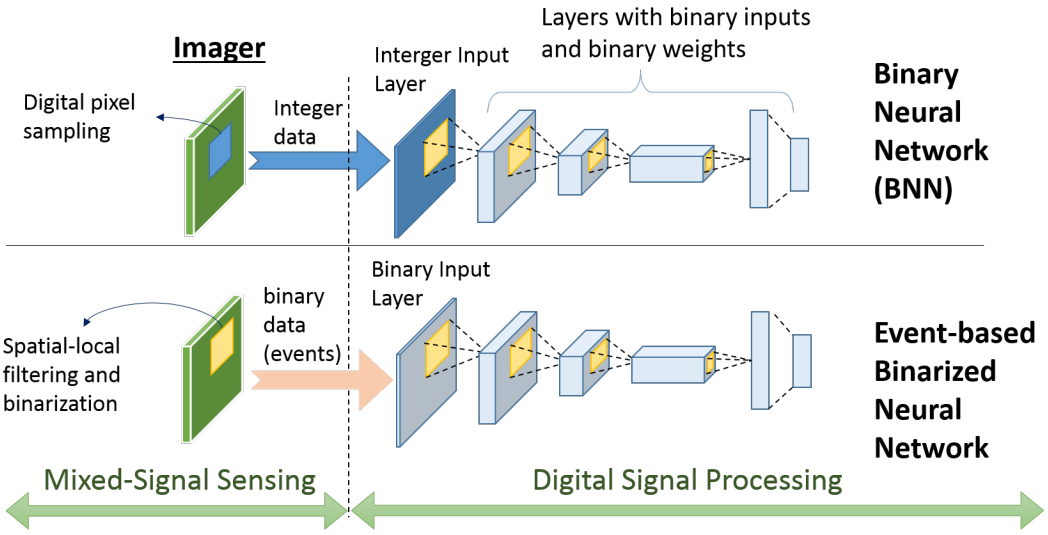


FIGURE 5.2: Comparison between a traditional BNN flow and the proposed Event-based BNN scheme, which exploits focal-plane processing for in-sensor binarization.

5.3 Event-based BNN

The Event-based Binarized Neural Network scheme is depicted in Figure 5.2. The presented approach combines the paradigm of event-based sensing with the new concept of Binarized Neural Networks (BNNs). If compared with traditional BNN architectures that operate on 3-channels RGB images, the Event-Based BNN is fed by sensor data that has been pre-processed and binarized on the image sensor die. This latter exploits a pixel-wise hardwired spatial filtering operation: a per-pixel mixed-signal circuit computes the weighted gradient across a neighboring pixel mask. Following this, gradient values are binarized by thresholding. This mixed-signal process can be seen as embedding the first binary convolution of the inference pipeline on the sensor die because it consists of binarizing the output of a local spatial gradient (illustrated in Figure 3.2).

The BNN is implemented on the digital signal processor and refers to the model presented by Courbariaux et al. [29]. Thanks to the input and weight binarization, any convolution reduces to a logic XNOR and a bit counting operation, resulting suitable to be implemented on ultra-low-power processors based on RISC architectures.

A preliminary experiment to assess the capability of the approach is conducted by benchmarking against the CIFAR-10 dataset [61]. To simulate the in-sensor processing of the event-based imager, a basic sensor model is used to convert RGB images of the dataset into the binary representation space. As a first approximation, the gradient contrast V_{EDGE} is computed as:

$$V_{EDGE} = \frac{\max(|p_E - p_O|, |p_N - p_O|)}{\max(p_E, p_O, p_N)} \tag{5.6}$$

where p_X is the greyscale value of the correspondent RGB pixel value. The binarization is formulated as:

$$V_O = sign(V_{EDGE} - V_{th}) \tag{5.7}$$

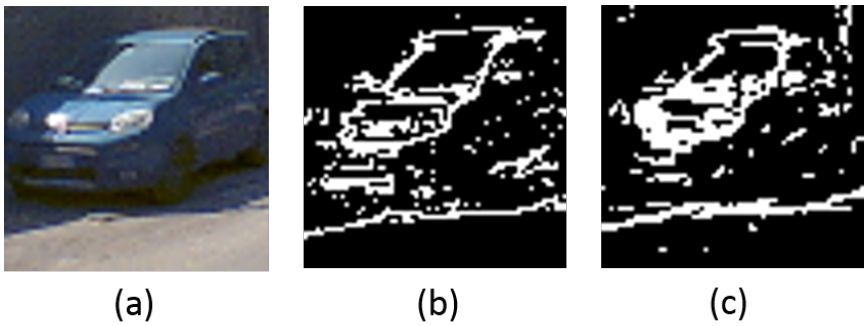


FIGURE 5.3: Image of a car taken with an RGB sensor (a) and with the binary imager (c). Image (b) is obtained by applying the transformation of Eq. 5.6-5.7 on the left RGB image to simulate the in-sensor processing.

TABLE 5.1: Accuracy on CIFAR10 dataset

Model	Accuracy
CNN with RGB input	91.36%
BNN with RGB input	86.78%
CNN with binarized input	72.50%
Event-Based BNN	68.94%

Figure 5.3 illustrates the qualitative result when transforming an RGB image into the representation space of the binary imager. The images labeled with (a) and (c) are captured respectively with an RGB image sensor and with the Event-Based sensor, while figure (b) is obtained by applying the transformation of equations (5.6)-(5.7) on the RGB image.

A VGG-like [98] BNN model is trained either with the original RGB data of the CIFAR dataset and with binarized data obtained from the transformation 5.6-5.7 of the original data space. The model is composed of 12 convolutional layers and 2 fully-connected layers and is trained following the approach of [29]. Table 5.1 lists the accuracy on the test set composed of 10k samples picked from the CIFAR-10 dataset. The evaluation also includes a baseline floating point CNN model with the same VGG topology, trained with RGB and binarized data. The results show that the specific kind of imager binarization leads to an 18.8% performance drop with respect to the baseline, while the additional binarization of the model leads to a further 3.5% reduction. Despite the not-negligible accuracy degradation, the training process actually leads to model convergence and hence can be exploited for training event-based binarized neural networks for an application-specific scenario. Section 5.5 shows that the accuracy degradation significantly attenuates by considering a reduced-complexity classification scenario, which is a more typical use-case for an always-on sensing front-end.

5.4 Implementation

The BNN software implementation is based on the code architecture presented in [78]. The proposed implementation aims at reading single binary values belonging to every receptive field and performs a 2D convolution and popcount. After accumulating the popcount results to the 3D convolution output (Eq. 5.3), the value is cast to floating point for the normalization and then thresholded, following Equation

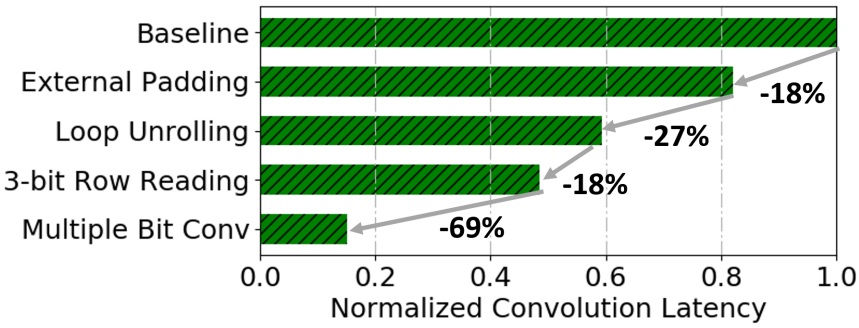


FIGURE 5.4: Average execution time to run a 3x3 convolution kernel for several optimization steps and their relative gains.

5.4. Several optimizations have been performed on the original code version to accelerate the computation and, at the same time, minimize the memory footprint. As major improvements, (a) the binary weights needed to produce an output layer are stored contiguously in memory to minimize the memory requirement instead of storing a 3x3 binary filter into a 3 bytes format, (b) the XNOR operation is performed on 32-bit data registers, to fully exploit the datapath length and (C) thanks to the convolution formulation expressed by Equation 5.5, the proposed implementation does not require any floating-point operation, with the exception of the last fully-connected layer, whose workload is however negligible in reference to the rest of the network inference task.

Memory Management. An BNN is implemented on the ultra-low power 4-core platform PULPv3 [96], described in Section 3.2.2. The BNN weight parameters are permanently stored in the L2 memory, besides the code region and the data input memory space. L1 memory serves for temporary storing of input and output data from the network layers. To this aim, two memory regions are sized as the maximum layer output capacity in the network, which is 2KB in case of the BNN topology of Table 5.2. At run-time, each core transfers a weight bank needed to produce a single output layer to a private L1 memory buffer, which has a size of 72 Bytes in this case.

Code Optimization and Parallelization. To optimize the BNN computation, the inference time of the model described in Tab. 5.2 is measured by running the BNN software on the instruction-accurate simulator of the parallel platform. Note that the BNN input is also binary, hence requiring only 1KB for transfer and storage, as opposed to the 12KB required by the system based on a RGB camera. When running on a single core, the baseline implementation spends the 96% of the computation on the convolution kernels of Equation 5.3, suggesting that this is the part that can benefit more from intensive optimization. For any output pixels, the mentioned operation consists of loading IFx3x3 binary input pixels and compress them in 32-bit registers, to be *xnored* with the correspondent weights before the popcount.

Fig. 5.4 illustrates the performance gain achieved by performing the optimization steps described below. The plot reports the average time to perform a 3x3 convolution, normalized with respect to the baseline. A first 18% cycles reduction is reached by padding every input layer before the convolution, hence avoiding to check border conditions in the inner loop. As a side effect, the L1 storage buffer has to be re-sized to contain the padded input (+13% memory for the considered network topology). An additional 27% reduction is achieved by exploiting loop unrolling. Doing this implies coding separately convolutional layers with different

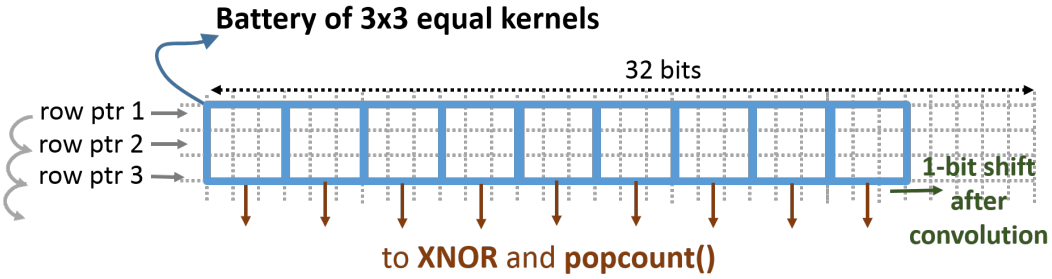


FIGURE 5.5: Optimized binary convolution kernel applied on every image plane. Three binary image rows are loaded and aligned with a battery of 3x3 convolution kernels. The masked popcount of the xored inputs resulting from any kernel position is accumulated and then the input is shifted to account all the possible input-filter alignment.

kernel size. For VGG-like models, which makes use only of 3x3 kernels, this is however not an issue. Others topologies featuring layers with different kernel sizes will pay an increased memory code footprint.

Going further, the implicit bit-level parallelism of data can be exploited by reading multiple input bits within a single load instruction. Thanks to this approach, any convolution of Equation 5.3 requires only IFx3 readings because a 3-bit row can be loaded by using a single load operation. This allows saving an additional 18% of the computation time.

To fully exploit the bit-level parallelism, the convolution operator can be applied to separate image input channels. The popcount results are accumulated along all the input channels before the binarization. Each image plane is tiled and scanned along the vertical direction. The spatial data order is exploited to load and analyze in parallel 32 binary pixels that belong to the same row. A battery of 9 identical binary filters is aligned to the loaded rows, by replicating the 2D weight filters, and xored. Figure 5.5 illustrates the process. For any of the filter masks, the popcount result from the several rows is accumulated and the input is shifted. After this, the row pointers are increased by 1 up to reach the bottom line. The process repeats on the next vertical tiles for every input channel. A significant 69% cycles reduction is achieved thanks to this strategy. From a memory viewpoint, an accumulation L1 memory space is required, sized as the maximum input spatial layer dimension (64x64 in this case).

On the considered digital processing platform, the BNN computation is parallelized over the 4-cores by dividing the workload along the output feature dimension. This contributes to speed-up the code execution by 3.88x, which is close to theoretical maximum 4x.

5.5 Evaluation

Classification accuracy The Event-Based BNN is evaluated with a real-life dataset, tailoring an always-on monitoring application for visual systems. A dual-camera setup, which includes a commercial RGB camera and the binary imager, is used for collecting 64x64 images, each one belonging to one of the three following categories: cars, cyclists and pedestrians. The acquisition system synchronizes the data capture of the two sensors, which are physically aligned to match their fields of view. Two

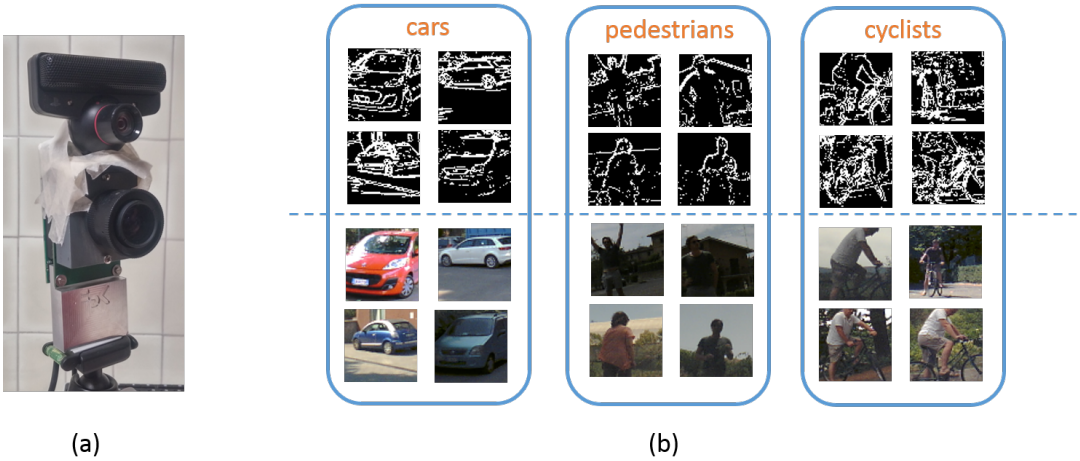


FIGURE 5.6: (a) dual-camera experimental setup (b) image samples corresponding to three different classes.

datasets for testing purpose are collected, one with RGB images and one with binary gradient images, containing 100 samples per class each. Figure 5.6 illustrates the dual camera experimental setup (a) and some examples of the acquired data (b).

A binarized VGG-like model is trained to classify the acquired images. The network is composed of 5 convolutional layers and 2 fully-connected layers (for a total of 23Mop/img). Table 5.2 reports the network parameters layer by layer. A pooling layer is placed after every convolution. The presented topology is defined to fit the memory requirements of the smart camera architecture. In total, the memory footprint required by this model is less than 20KB. As a baseline for the proposed event-based BNN, a binarized neural network with the same topology is trained and tested on 8-bit RGB data.

Both the BNN models are trained using Torch [26] for 100 epochs using the adaMax shift-based version proposed by [29] and a batch size of 128. Learning rate is set to 0.01 and divided by 10 every 15 epochs. To increase the generalization of the training process, the training and validation datasets are built by combining labeled patches from the KITTI [45] and MIO-TCD [105] datasets. Training data is augmented with random rotation to increase the number of training samples up to about 60k. The validation set is composed of 900 unique samples. When training the event-based BNN, training and validation data are binarized with Equations 5.6-5.7. Trained models with the highest accuracy on the validation dataset report an accuracy on the real-life testing data of 84.6% and 81.6%, for RGB and gradient binarized input scenario, respectively. Therefore, the event-based BNN architecture presents a contained performance drop of 3% over the 3-classes application scenario.

Energy Evaluation. To assess the energy efficiency of the proposed solution, the event-based binary visual node is benchmarked against a baseline system featuring a state-of-the-art low-power RGB imager [52] and the same processing unit, which runs a BNN with 8-bit 3 channels data inputs.

[1]Conv3x3(x,y) is a convolutional layer with 3x3 weight filter size, x input layers and y output layers, FC(x, y) is a fully connected layer with x input neurons and y output neurons, #ich=3 for RGB input and #ich=1 for binary input

[2]The imager features a 324×244 resolution with Bayer color filter map, which roughly corresponds to a 3-channel QQVGA resolution

TABLE 5.2: VGG-like BNN Model

Model Topology [1]
Conv3x3(#ich, 16) + MaxPool2x2
Conv3x3(16, 32) + MaxPool2x2
Conv3x3(32, 48) + MaxPool2x2
Conv3x3(48, 64) + MaxPool2x2
Conv3x3(64, 96) + MaxPool2x2
FC(384, 64)
FC(64,3)

TABLE 5.3: Event Based BNN Energy Comparison

Scenario	**BNN with RGB input**	**Event-based BNN**
Image Sensor Power Consumption	1.1mW @30fps	100μW @50fps
Image Size	324×244 (617kbits) [2]	128×64 (8kbits)
Image Sensor Energy for frame capture	66.7 μJ	2 μJ
Transfer Time (4bit SPI @50MHz)	3.1 msec	0.04 msec
Transfer Energy (8.9mW @0.7V)	28 μJ	2 μJ
BNN Execution Time (168MHz)	81.3 msec	75.3 msec
BNN Energy consumption (8.9mW @0.7V)	725 μJ	671 μJ
Total System Energy for Classification	820 μJ	674 μJ

Table 5.3 reports the energy comparison between the event-based BNN and the baseline system for image acquisition and classification. It includes the contribution of the imager, sensor-to-processor data transfer and the 4-core processor that runs the binarized network. The active power of the processing unit is measured as 8.9mW when operating at 168MHz with a voltage supply of 0.7V. The platform is kept in the active state during data transfer and BNN computation. Because of the higher amount of input data (a total of 324×244 8-bit pixels [52] instead of a single 128x64 binary channel), the baseline scenario features a 77.2x higher data transfer time. Furthermore, the event-based BNN shows a smaller BNN computation time by 7.4% thanks again to the reduced amount of input data. Given all these contributions, the event-based BNN scenario reports a system-level energy reduction of 17.8% with respect to the baseline.

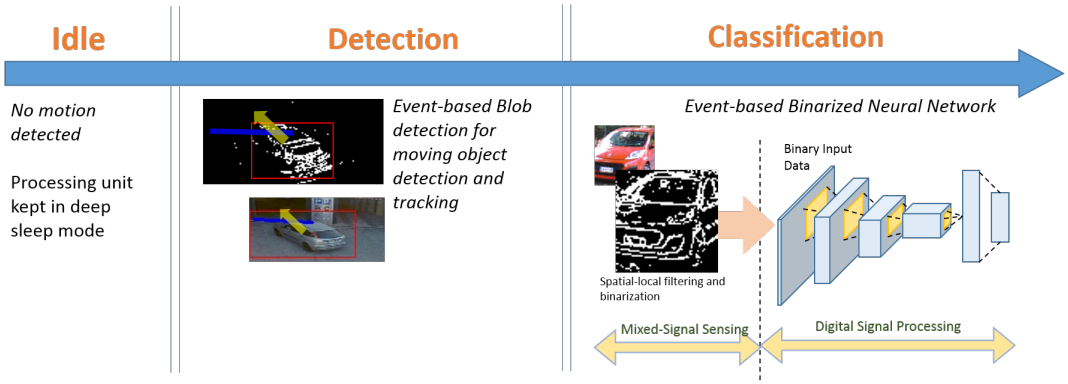


FIGURE 5.7: Event-Based Binarized Neural Network analysis flow applied to always-on monitoring applications.

TABLE 5.4: Event-Based vs Frame-based

Statistics per frame	Frame-Based	Event-based
Idle (no motion)		
Sensor Power	1.1mW (grey)	20μW
Avg Sensor Data	19764 Bytes	-
Transfer Time	790μsec	-
Processing Time	3.02 msec	-
Avg Processor Power	1.45mW	0.3mW (sleep)
Detection		
Sensor Power	1.1mW (grey)	60μW
Avg Sensor Data	19764 Bytes	~536 Bytes
Transfer Time	790μsec	21.4μsec
Processing Time	3.47 msec	187.6μsec
Avg Processor Power	1.57mW	0.511mW
Classification		
Sensor Power	2mW (RGB)	60μW
Avg Sensor Data	79056 Bytes	1024 Bytes
Transfer Time	3.16 msec	41μsec
Processing Time	81.3 msec	75.3 msec
Processor Energy	760 μJ	677 μJ

Event-Driven Sensing Evaluation. As highlighted in Chapter 4, a design methodology relying on an event-based computational model leads to major energy-efficiency improvements over state of the art solutions. When dealing with always-on monitoring applications, the optimized BNN classification engine can be run on top of the triggering process to leverage the benefits of the Event-Based Smart Visual Systems. Therefore, the system enhances its recognition capabilities by assigning a unique label to a detected interesting event. Figure 5.7 illustrates the working flow of the proposed scheme. As a case-study, a *parking entrance monitoring* scenario is targeted, where an alert signal is triggered when a moving object (a car) enters the parking gate while the system is kept in sleep mode, but still working, if no motion is detected. The BNN execution starts once an alert is generated to classify it. In the following, a comparison between the proposed Event-Based BNN scheme and a traditional frame-based system is provided.

In the analyzed case, the sensing activity is driven by moving cars passing in front of the camera. Any of them generates 2.7 seconds of data recording. Tab. 5.4 reports the average statistics to acquire and elaborate the signals on both the event-based and the frame-based scenario, along with the different energy costs for any of the following phases: (i) *idle*, when no motion is detected by the mixed-signal image processing because of a static background, (ii) *detection*, aiming at generating alert signals, and (iii) *classification* upon detection, which implies transferring data and running the BNN classifier. Concerning the *idle* and *detection* phases, the energy cost is expressed in terms of average power consumption. A frame rate of 30fps is considered, along with the usage of duty cycling to reduce the energy consumption of the digital processor. When in the sleep state, the processing unit consumes 0.3mW @0.7V, due to the memory region that cannot be power-gated because of data-retention. The event-based system keeps the digital platform in the sleep state during the idle phase, due to the low amount of generated events. Within the detection phase, a higher sensor datarate causes the processor to wake up for data processing. The power costs are still contained because of the limited number of events to be transferred and elaborated. Once a relevant event is detected through event-driven processing of Section 3.3, the classification task is triggered. On the

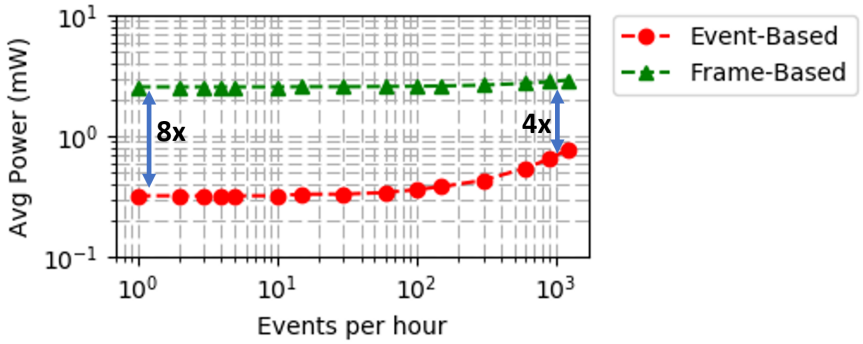


FIGURE 5.8: Average power consumption of event-based (in red) and frame-based (green) visual nodes with respect to a varying external event rate within an always-on monitoring application.

contrary, within the frame-based scenario, the sensor always transfers image data to the processor to determine the presence of relevant objects either on the idle or the detection phase. Hence, these phases are characterized by a similar power cost. The data analytic flow includes the background subtraction, morphological filtering and the extraction of the connected components. During the detection phase, the workload slightly increases due to the additional Kalman filtering and triggering process. As a data source, the image sensor provides QQVGA greyscale 8-bit data at a power cost of 1.1mW [52].

Fig. 5.8 shows the average power consumption for a varying number of moving objects per hour accessing to the parking gate. On the frame-based system the power is weakly dependent on the event frequency, while the event-based system presents an increasing power cost due to the increased activation rate of the digital processor at a higher event rate. When the number of event per hour increases, the event-based visual system presents an energy saving of 4x, which can raise up to 8x at a reduced event activity. Moreover, if the context activity tend to decrease, the system average power consumption is kept as low as the sleep power of $300\mu W$.

5.6 Summary

This Chapter presented the concept of Event-Based Binarized Neural Networks, which couples together the energy benefits of an Event-Driven Sensing and Processing approach with the visual performance of Binarized Neural Networks (BNNs). The Event-Based imager provides binarized data as an input of the BNN, which is implemented in an efficient way on a parallel digital engine. If compared with a baseline systems featuring an RGB imager, the proposed approach presents a power consumption reduced by 17.8% due to the in-sensor early-computation. Moreover, when considering an always-on monitoring application, the BNN can be triggered by the detection of moving objects. In this case, thanks to the Event-Based visual system, the average power consumption can be up to 8x lower than the one of the baseline system in case of infrequent detections, while reaching an average power consumption as low as the sleep power of 0.3mW.

Chapter 6

Conclusion

The Internet of Things revolution is nowadays driving the development of novel technologies for collecting and processing a massive amount of data. At the edge of the envisioned infrastructure, a plethora of interconnected and autonomous devices is expected to form a widespread sensing layer, that transmits sensed data through the network to other nodes or to a central system. However, due to the stringent energy requirements, the design process of the end-node devices results extremely challenging. In the context of visual sensing, the high power cost of most diffused smart camera devices cannot be sustained by battery-powered autonomous devices. In fact, a camera-based system typically includes power-hungry components to perform visual data sampling and for running computationally demanding computer vision tasks.

The thesis work describes novel design strategies and techniques to enable a smart ultra-low power visual sensing at the edge of the IoT network. As a design target, *this work aims at building a smart and flexible visual sensor node, which achieves the detection performance of leading-edge computer vision models but under the energy constraints of autonomous devices, i.e. a power cost within the ambitious micro-watt range*. To reach the goal, the following challenges have been addressed.

- **HW-SW Co-Design.** Edge devices require high-energy efficiency but high computational power to handle complex computational vision models. Concerning the HW, this implies a design effort either on the technology side but also at the architectural level. Optimized programmable engines for data analytics enable an efficient computation and provide the needed flexibility to ease the design process. Associated to this, the software implementation needs to be oriented toward energy minimization and therefore able to leverage the underlying HW architecture in an efficient way.

- **System Integration.** From a system level point of view, the components' operations need to be coordinated to minimize the energy wastes. To this purpose, a system power management strategy must be designed and implemented, such that controls the system operating modes without degrading the sensing quality.

- **Data Analytics.** The computer vision algorithm design needs to be aware of the architectural resource constraints. An optimal data analytics flow aims at minimize the energy consumption while presenting high detection capabilities.

The thesis work addressed the mentioned challenges by proposing an Ultra-Low Power Event-Based smart camera design, which relies on novel design techniques such as (a) focal plane processing and (b) event-driven sensing and computation.

According to the first one, part of the visual processing is moved on the sensor die by exploiting mixed-signal processing circuits. This enables an efficient early-extraction of low-level visual features and brings the power cost down to the μW range. Thanks to the usage of mixed-signal circuits, a smart camera architecture can shift the sensing paradigm from the traditional frame-based to a bio-inspired event-based flavor. In this scenario, the image sensor triggers illumination changes and dispatch them to the progressing unit, in the form of *Events*. To anaylize visually relevant data, the digital processing engine features a more lightweight computation than traditional frame-based architectures.

Concerning the aforementioned challenges:

- Chapter 3 described the HW-SW co-design of a smart camera architecture coupling a parallel processor with a mixed-signal image sensor. This latter embeds focal plane processing capabilities and address-event readout. The digital computation is applied to the generated visual events, leading to a 57x processing energy reduction with respect to a traditional frame-based computation over a monitoring application. Moreover, thanks to the parallel engine and the ultra-low power event-based camera, the overall system energy cost results to be two order of magnitude lower than a baseline system featuring off-the-shelf components.

- Chapter 4 showed a power-optimized system integration and the design of a specialized camera interface to enable the implementation of a context-aware system power management strategy. Differently from other smart camera designs, the power management strategy, which controls the components operating modes, depends on the motion detected on the camera field of view. Such an event-driven computational model is enabled by means of the event-based image sensor and the specification of the camera interface, which handles the sensor operation but also drives the power control strategy. If compared with other proposed camera-based system, the proposed implementation achieves more than a 10x lower average power consumption.

- Chapter 5 described the implementation of a Binarized Neural Network for classification purposes on the Event-Based mixed-signal visual system. The implementation exploits both the analog processing and the computational power of the digital parallel engine. The BNN model, which reduces the bit-precision of a convolutional neural network to a single bit to favor the deployment on a resource-constrained device, is implemented such as to leverage the maximum computational capabilities offered by the parallel platform. The system shows a processing time of 75 msec for the inference task and an energy consumption reduced by 17.8% with respect to a baseline system featuring an RGB image sensor. Moreover, when the BNN computation is triggered by the event-driven object detection flow, the average power consumption can be up to 8x lower than the one of the baseline system. In case of infrequent detections, the Ultra-Low Power Smart Visual system achieves a power cost as low as the sleep power of 0.3mW.

Based on the presented results, the design techniques discussed within the thesis have demonstrated improved energy characteristics with respect to state-of-the-art solutions, showing a viable way to enable microwatt vision technologies at the edge of the Internet of Things infrastructure but featuring a visual performance quality not degraded with respect to more diffused computer vision models.

6.1 Future Directions

The thesis work discussed the implementation of novel techniques for ultra-low power visual sensing. By leveraging a mixed-signal event-based sensing and processing architecture, the energy-efficiency of smart camera systems raises up to match the energy requirement of battery-powered IoT devices. Thanks to this, new opportunities arise for vision-based applications, which are currently limited by power-hungry devices. Differently from the actual state, the presented visual technology enables the placement of autonomous devices within the environment, featuring smart capabilities for early detection and triggering of interesting events. However, due to the novelty of this design techniques, tools and design methodologies are still in early stage and need more work to gain higher maturity and robustness. This issue needs to be addressed in the upcoming future to make the design process more solid.

To favor the design automation of mixed-signal visual systems, the HW components are expected to feature a higher flexibility than current available samples. As an example, the early-extraction of visual features through mixed signal-circuits can include some extra knobs to realize multiple and even more complex parametric filtering on the sensor side. This will eases the mapping of visual data analytics at the pixel level in an energy efficient way. More in general, new architectural solutions can investigate an optimal partition of the visual workload between analog and digital processing together with the programmability level of early-processing circuits. For instance, an image sensor could also embeds more complex operating modes and decision rules to trigger the wake up of the digital part, without loosing generality. Along with it, the HW design must consider the trade off between flexibility and power. Featuring highly-flexible devices commonly implies a lower energy efficiency and viceversa. This should be strictly considered when designing future enabling technologies for the IoT domain.

At the system level, novel tools should be tailored to automate the application development into resource-constrained mixed-signal architectures, also taking into account the energy requirements. This includes the specification of novel HW-SW co-design flows, which consider both the analog filtering and the SW digital processing. To fully exploit the HW capabilities, the algorithm design flow for data analytics should be part as well of the system level design process. Currently, the design of computer vision algorithms for constrained devices mostly relies on approximation techniques of leading strategies (e.g. the case of BNNs). In contrast, the development of more sophisticated models should take into account the sensor data generation process and the underlying HW architecture. For instance, in the context of event-driven processing, a general end-to-end framework would be extremely beneficial for designing data analytic flows, depending on the mixed-signal architecture, and, at the same time, match the requirement of the visual applications enabled by the novel proposed design techniques.

Bibliography

[1] In: Cypress Semiconductors - S25FS Non-Volatile Memory - Datasheet.

[2] Kevin Abas, Caio Porto, and Katia Obraczka. "Wireless smart camera networks for the surveillance of public spaces". In: *Computer* 47.5 (2014), pp. 37–44.

[3] Massimo Alioto. "IoT: Bird's Eye View, Megatrends and Perspectives". In: *Enabling the Internet of Things*. Springer, 2017, pp. 1–45.

[4] Massimo Alioto. "Ultra-low power VLSI circuit design demystified and explained: A tutorial". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 59.1 (2012), pp. 3–29.

[5] Stefano Alletto et al. "An indoor location-aware system for an IoT-based smart museum". In: *IEEE Internet of Things Journal* 3.2 (2016), pp. 244–253.

[6] *Ambiq Apollo Ulta Low Power Microcontroller*. Accessed: 2016-07-30. URL: http://ambiqmicro.com/.

[7] *ARM Cortex-M Series*. Accessed: 2016-07-30. URL: https://www.arm.com/products/processors/cortex-m.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[9] Lorenzo Baraldi et al. "Gesture recognition using wearable vision sensors to enhance visitors' museum experiences". In: *IEEE Sensors Journal* 15.5 (2015), pp. 2705–2714.

[10] Ahmed Nabil Belbachir. *Smart cameras*. Springer, 2010.

[11] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. "A survey of design techniques for system-level dynamic power management". In: *IEEE transactions on very large scale integration (VLSI) systems* 8.3 (2000), pp. 299–316.

[12] Raphael Berner et al. "Dynamic vision sensor for low power applications". In: *Consumer Electronics (ISCE 2014), The 18th IEEE Int. Symposium on*. IEEE. 2014, pp. 1–2.

[13] *Blackfin Low Power Imaging Platform (BLIP)*. Accessed: 2016-07-30. URL: http://www.analog.com/en/index.html.

[14] Alessio Botta et al. "Integration of cloud computing and internet of things: a survey". In: *Future Generation Computer Systems* 56 (2016), pp. 684–700.

[15] Christian Brandli et al. "A 240× 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor". In: *IEEE Journal of Solid-State Circuits* 49.10 (2014), pp. 2333–2341.

[16] Stephen J Carey, David RW Barr, and Piotr Dudek. "Low power high-performance smart camera system based on SCAMP vision sensor". In: *J. of Systems Architecture* 59.10 (2013), pp. 889–899.

[17] M. Casares et al. "Energy-efficient feedback tracking on embedded smart cameras by hardware-level optimization". In: *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras*. 2011, pp. 1–6. DOI: 10.1109/ICDSC.2011.6042915.

[18] *Centeye Stonyman/Hawksbill silicon documentation*. Accessed: 2016-07-30. URL: http://www.centeye.com/.

[19] Feng Chen et al. "Data mining for the internet of things: literature review and challenges". In: *International Journal of Distributed Sensor Networks* 11.8 (2015), p. 431047.

[20] Huaijin G Chen et al. "ASP vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels". In: *Proc. IEEE CVPR*. 2016, pp. 903–912.

[21] Phoebus Chen et al. "A low-bandwidth camera sensor platform with applications in smart camera networks". In: *ACM Trans. on Sensor Networks (TOSN)* 9.2 (2013), p. 21.

[22] Thou-Ho Chen, Yu-Feng Lin, and Tsong-Yi Chen. "Intelligent vehicle counting method based on blob analysis in traffic surveillance". In: *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*. IEEE. 2007, pp. 238–238.

[23] Yen-Kuang Chen. "Challenges and opportunities of internet of things". In: *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. IEEE. 2012, pp. 383–388.

[24] Jaehyuk Choi et al. "A 3.4-μW Object-Adaptive CMOS Image Sensor With Embedded Feature Extraction Algorithm for Motion-Triggered Object-of-Interest Imaging". In: *IEEE Journal of Solid-State Circuits* 49.1 (2014), pp. 289–300.

[25] *CMUcam: Open Source Programmable Embedded Color Vision Sensors*. http://www.cmucam.org/. Accessed: 2016-07-30.

[26] R. Collobert, S. Bengio, and J. Mariéthoz. *Torch: a modular machine learning software library*. Tech. rep. Idiap, 2002.

[27] Francesco Conti et al. "Accelerated visual context classification on a low-power smartwatch". In: *IEEE Transactions on Human-Machine Systems* 47.1 (2017), pp. 19–30.

[28] Francesco Conti et al. "An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.9 (2017), pp. 2481–2494.

[29] M. Courbariaux et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1". In: *arXiv preprint arXiv:1602.02830* (2016).

[30] Tobi Delbruck. "Frame-free dynamic digital vision". In: *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*. 2008, pp. 21–26.

[31] Tobi Delbruck. "Neuromorophic vision sensing and processing". In: *Solid-State Device Research Conference (ESSDERC), 2016 46th European*. IEEE. 2016, pp. 7–14.

[32] Tobi Delbruck and Manuel Lang. "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor". In: *Frontiers in neuroscience* 7 (2013).

[33] Oscar Deniz et al. "Eyes of Things". In: *Sensors* 17.5 (2017), p. 1173.

[34] Rodney J Douglas and Kevan AC Martin. "Recurrent neuronal circuits in the neocortex". In: *Current biology* 17.13 (2007), R496–R500.

[35] P Dudek and SJ Carey. "General-purpose 128× 128 SIMD processor array with integrated image sensor". In: *Electronics Letters* 42.12 (2006), pp. 678–679.

[36] Ericsson. *Networked Society City Index, White Paper*. Accessed: 2016-07-30. 2016. URL: https://www.ericsson.com/assets/local/networked-society/reports/city-index/2016-networked-society-city-index.pdf.

[37] Jorge Fernandez-Berni, Ricardo Carmona-Galán, and Ángel Rodríguez-Vázquez. "FLIP-Q: A QCIF resolution focal-plane array for low-power image processing". In: *Low-Power Smart Imagers for Vision-Enabled Sensor Networks*. Springer, 2012, pp. 67–109.

[38] Jorge Fernández-Berni, Ricardo Carmona-Galán, and Ángel Rodríguez-Vázquez. "Vision-enabled WSN Nodes: State of the Art". In: *Low-Power Smart Imagers for Vision-Enabled Sensor Networks*. Springer, 2012, pp. 5–20.

[39] Jorge Fernández-Berni et al. "Focal-plane sensing-processing: A power-efficient approach for the implementation of privacy-aware networked visual sensors". In: *Sensors* 14.8 (2014), pp. 15203–15226.

[40] Jorge Fernández-Berni et al. "Wi-FLIP: A wireless smart camera based on a focal-plane low-power image processor". In: *Distributed Smart Cameras (ICDSC), 2011 5th ACM/IEEE Int. Conf. on*. IEEE. 2011, pp. 1–6.

[41] Philippe Flatresse et al. "Ultra-wide body-bias range LDPC decoder in 28nm UTBB FDSOI technology". In: *Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), 2013 IEEE Int.* IEEE. 2013, pp. 424–425.

[42] Guillermo Gallego et al. "Event-based, 6-DOF camera tracking for high-speed applications". In: *arXiv preprint arXiv:1607.03468* (2016).

[43] Leonardo Gasparini et al. "An ultralow-power wireless camera node: Development and performance analysis". In: *Instrumentation and Measurement, IEEE Transactions on* 60.12 (2011), pp. 3824–3832.

[44] Michael Gautschi et al. "Tailoring instruction-set extensions for an ultra-low power tightly-coupled cluster of OpenRISC cores". In: *Very Large Scale Integration (VLSI-SoC), 2015 IFIP/IEEE Int. Conf. on*. IEEE. 2015, pp. 25–30.

[45] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[46] D Giovanelli, B Milosevic, and E Farella. "Bluetooth Low Energy for data streaming: Application-level analysis and recommendation". In: *Advances in Sensors and Interfaces (IWASI), 2015 6th IEEE International Workshop on*. IEEE. 2015, pp. 216–221.

[47] Massimo Gottardi, Nicola Massari, and Syed Arsalan Jawed. "A 100μ W 128×64 Pixels Contrast-Based Asynchronous Binary Vision Sensor for Sensor Networks Applications". In: *IEEE Journal of Solid-State Circuits* 44.5 (2009), pp. 1582–1592.

[48] Harmke de Groot. "IoT and the cloud: A hacked personality and an empty battery head-ache or an intuitive environment to make our lives easier?" In: *SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. IEEE. 2015, pp. 1–5.

[49] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[50] S. Hanson et al. "A 0.5 V Sub-Microwatt CMOS Image Sensor With Pulse-Width Modulation Read-Out". In: *IEEE Journal of Solid-State Circuits* 45.4 (2010), pp. 759–767. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2040231.

[51] Stephan Hengstler et al. "MeshEye: A Hybrid-resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance". In: *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*. IPSN '07. Cambridge, Massachusetts, USA: ACM, 2007, pp. 360–369. ISBN: 978-1-59593-638-7. DOI: 10.1145/1236360.1236406. URL: http://doi.acm.org/10.1145/1236360.1236406.

[52] *Himax HM01B0 sensor*. Accessed: 2016-07-30. URL: http://www.himax.com.tw/.

[53] Gartner Inc. *The Internet of Things Is a Revolution Waiting to Happen*. Accessed: 2016-07-30. 2015. URL: http://www.gartner.com/smarterwithgartner/the-internet-of-things-is-a-revolution-waitingto-happen.

[54] D INFSO. "Networked Enterprise & RFID INFSO G. 2 Micro & Nanosystems, in co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future [R]". In: *Information Society and Media, Tech. Rep* (2008).

[55] Hrishikesh Jayakumar et al. "Powering the internet of things". In: *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM. 2014, pp. 375–380.

[56] *JeVois Smart Machine Vision Camera*. Accessed: 2016-07-30. URL: http://jevois.org/.

[57] Changhyeon Kim et al. "An ultra-low-power and mixed-mode event-driven face detection SoC for always-on mobile applications". In: *ESSCIRC 2017-43rd IEEE European Solid State Circuits Conference*. IEEE. 2017, pp. 255–258.

[58] Gyouho Kim et al. "A 467nW CMOS visual motion sensor with temporal averaging and pixel aggregation". In: *Proc. IEEE ISSCC*. 2013, pp. 480–481.

[59] T. Ko et al. "Exploring Tradeoffs in Accuracy, Energy and Latency of Scale Invariant Feature Transform in Wireless Camera Networks". In: *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*. 2007, pp. 313–320. DOI: 10.1109/ICDSC.2007.4357539.

[60] Gerd Kortuem et al. "Smart objects as building blocks for the internet of things". In: *IEEE Internet Computing* 14.1 (2010), pp. 44–51.

[61] A. Krizhevsky and G. Hinton. "Learning multiple layers of features from tiny images". In: (2009).

[62] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[63] Purushottam Kulkarni et al. "SensEye: a multi-tier camera sensor network". In: *Proc. of the 13th annual ACM int. conf. on Multimedia*. ACM. 2005, pp. 229–238.

[64] Xavier Lagorce et al. "Asynchronous event-based multikernel algorithm for high-speed visual features tracking". In: *IEEE transactions on neural networks and learning systems* 26.8 (2015), pp. 1710–1720.

[65] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[66] Edward H Lee and S Simon Wong. "Analysis and Design of a Passive Switched-Capacitor Matrix Multiplier for Approximate Computing". In: *IEEE Journal of Solid-State Circuits* 52.1 (2017), pp. 261–271.

[67] Jun Haeng Lee et al. "Real-time gesture interface based on event-driven processing from stereo silicon retinas". In: *IEEE transactions on neural networks and learning systems* 25.12 (2014), pp. 2250–2263.

[68] Juan Antonio Leñero-Bardallo, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. "A 3.6*mu*s Latency Asynchronous Frame-Free Event-Driven Dynamic-Vision-Sensor". In: *IEEE Journal of Solid-State Circuits* 46.6 (2011), pp. 1443–1455.

[69] Chenghan Li et al. "Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor". In: *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. IEEE. 2015, pp. 718–721.

[70] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128x128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE journal of solid-state circuits* 43.2 (2008), pp. 566–576.

[71] Robert LiKamWa et al. "Draining our glass: An energy and heat characterization of google glass". In: *Proceedings of 5th Asia-Pacific Workshop on Systems*. ACM. 2014, p. 10.

[72] Robert LiKamWa et al. "Energy characterization and optimization of image sensing toward continuous mobile vision". In: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM. 2013, pp. 69–82.

[73] Robert LiKamWa et al. "RedEye: analog ConvNet image sensor architecture for continuous mobile vision". In: *Proc. IEEE ISCA*. 2016, pp. 255–266.

[74] M Litzenberger et al. "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor". In: *Digital Signal Processing Workshop, 12th-Signal Processing Education Workshop, 4th*. IEEE. 2006, pp. 173–178.

[75] Martin Litzenberger et al. "Embedded smart camera for high speed vision". In: *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on*. IEEE. 2007, pp. 81–86.

[76] Igor Loi et al. "Exploring multi-banked shared-L1 program cache on ultra-low power, tightly coupled processor clusters". In: *Proc. of the 12th ACM Int. Conf. on Computing Frontiers*. ACM. 2015, p. 64.

[77] Michele Magno et al. "Multimodal video analysis on self-powered resource-limited wireless smart camera". In: *Emerging and Selected Topics in Circuits and Systems, IEEE J. on* 3.2 (2013), pp. 223–235.

[78] B. McDanel, S. Teerapittayanon, and H. Kung. "Embedded Binarized Neural Networks". In: *Proc. of 2017 Int Conf Embedded Wireless Systems and Networks* (2017).

[79] Carver Mead. "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[80] Paul A Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface". In: *Science* 345.6197 (2014), pp. 668–673.

[81] *Microsemi's IGLOO nano FPGA*. http://www.microsemi.com/. Accessed: 2016-07-30.

[82] Elias Mueggler, Basil Huber, and Davide Scaramuzza. "Event-based, 6-DOF pose tracking for high-speed maneuvers". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 2761–2768.

[83] Georg R Müller and Jörg Conradt. "A miniature low-power sensor system for real time 2D visual tracking of led markers". In: *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2429–2434.

[84] Saman Naderiparizi et al. "Glimpse: A Programmable Early-Discard Camera Architecture for Continuous Mobile Vision". In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '17. Niagara Falls, New York, USA: ACM, 2017, pp. 292–305. ISBN: 978-1-4503-4928-4. DOI: 10.1145/3081333.3081347. URL: http://doi.acm.org/10.1145/3081333.3081347.

[85] Saman Naderiparizi et al. "Ultra-low-power Wireless Streaming Cameras". In: *CoRR* abs/1707.08718 (2017). arXiv: 1707.08718. URL: http://arxiv.org/abs/1707.08718.

[86] *NanEye 2D sensor*. Accessed: 2016-07-30. URL: http://www.awaiba.com/product/naneye/.

[87] *OmniVison Image Sensors*. Accessed: 2016-07-30. URL: http://www.ovt.com/image-sensors/.

[88] *OpenMV, Machine Vision with Python*. Accessed: 2016-07-30. URL: https://openmv.io/.

[89] Ashwin Pajankar. *Raspberry Pi Computer Vision Programming*. Packt Publishing Ltd, 2015.

[90] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS". In: *IEEE Journal of Solid-State Circuits* 46.1 (2011), pp. 259–275.

[91] Arnab Raha and Vijay Raghunathan. "Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system". In: *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM. 2017, p. 74.

[92] Mohammad Rahimi et al. "Cyclops: in situ image sensing and interpretation in wireless sensor networks". In: *Proc. of the 3rd int. conf. on Embedded networked sensor systems*. ACM. 2005, pp. 192–204.

[93] M. Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European Conf. on Computer Vision*. Springer. 2016, pp. 525–542.

[94] Á Rodríguez-Vázquez et al. "In the quest of vision-sensors-on-chip: Pre-processing sensors for data reduction". In: *Electronic Imaging* 2017.11 (2017), pp. 96–101.

[95] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[96] Davide Rossi et al. "A Self-Aware Architecture for PVT Compensation and Power Nap in Near Threshold Processors". In: *IEEE Design & Test* 34.6 (2017), pp. 46–53.

[97] Rafael Serrano-Gotarredona et al. "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking". In: *IEEE Transactions on Neural Networks* 20.9 (2009), pp. 1417–1438.

[98] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[99] *STM32 32-bit ARM Cortex MCUs*. Accessed: 2016-07-30. URL: http://www.st.com/.

[100] Manuel Suárez et al. "Low-Power CMOS Vision Sensor for Gaussian Pyramid Extraction". In: *IEEE Journal of Solid-State Circuits* 52.2 (2017), pp. 483–495.

[101] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[102] Bulent Tavli et al. "A survey of visual sensor network platforms". In: *Multimedia Tools and Applications* 60.3 (2012), pp. 689–726.

[103] Thiago Teixeira, Andreas G Andreou, and Eugenio Culurciello. "Event-based imaging with active illumination in sensor networks". In: *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE. 2005, pp. 644–647.

[104] Thiago Teixeira et al. "Address-event imagers for sensor networks: evaluation and modeling". In: *Proceedings of the 5th international conference on Information processing in sensor networks*. ACM. 2006, pp. 458–466.

[105] "The Traffic Surveillance Workshop and Challenge 2017 (TSWC- 2017)". In: http://podoce.dinf.usherbrooke.ca.

[106] Jilles Vreeken. *Spiking neural networks, an introduction*. 2003.

[107] Andrea Zanella et al. "Internet of things for smart cities". In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32.