An Event-Driven Ultra-Low-Power Smart Visual Sensor

Manuele Rusci, Davide Rossi, Michela Lecca, Massimo Gottardi, Elisabetta Farella, and Luca Benini

***Abstract*—In this paper we present an ultra-low-power smart visual sensor architecture. A $10.6\mu W$ low-resolution contrast-based imager featuring internal analog pre-processing is coupled with an energy-efficient quad-core cluster processor that exploits near-threshold computing within a few mW power envelope. We demonstrate the capability of the smart camera on a moving object detection framework. The computational load is distributed among mixed-signal pixel and digital parallel processing. Such local processing reduces the amount of digital data to be sent out of the node by 91%. Exploiting context aware analog circuits, the imager only dispatches meaningful post-processed data to the processing unit, lowering the sensor-to-processor bandwidth by $31\mathrm{x}$ with respect to transmitting a full pixel frame. To extract high-level features, an event-driven approach is applied to the sensor data and optimized for parallel runtime execution. A $57.7\mathrm{x}$ system energy saving is reached through the event-driven approach with respect to frame-based processing, on a low-power MCU node. The near-threshold parallel processor further reduces the processing energy cost by $6.64\mathrm{x}$, achieving an overall system energy cost of $1.79\mu\mathrm{J}$ per frame, which results to be $21.8\mathrm{x}$ and up to $383\mathrm{x}$ lower than, respectively, an event-based imaging system based on asynchronous visual sensor and a traditional frame-based smart visual sensor.**

***Index Terms*—Smart Visual Sensor, Embedded System, Ultra-Low-Power, Event-Driven.**

I. Introduction

VISION is considered as one of the richest sources to explore and understand the surrounding world [1]. Processing visual signals is crucial for many applications such as video surveillance, traffic monitoring, people/object tracking, life assisted living. All these applications would benefit from always-on image sensors coupled with processing engines to extract relevant visual information from raw pixels. Traditional real-time image and video processing consist of computationally heavy tasks, typically requiring powerful computing devices (e.g. GP-GPUs, SIMD-capable high speed processor cores) and large memory footprint due to the massive amount of pixels coming from frame-based image sensors.

In the context of wireless sensor nodes, the power consumption of vision systems has to be scaled down according to the available energy supply resources (i.e., small batteries or harvesters) [2]. In this scenario, where low-power MCUs (e.g. ARM Cortex M) replace high-end embedded processors, wireless transmission becomes a major contributor to the system energy consumption [3]. To reduce the cost of wireless transmission, a system design strategy is to bring *intelligence* closer to the sensor. Such intelligent systems, referred here as *smart visual sensors*, not only acquire an image, but also perform visual processing on it, generating a high-level description of the observed scene. Hence, by exploiting local processing, the node is able to extract high-level features from the sensed data, and only relevant information are dispatched through the wireless channel [4]. These intelligent systems are usually composed of an image sensor with embedded processing and digital interface that allows the communication with an host system or with the user [1]. Energy-efficiency is a key feature to provide the required computational power.

Several low power smart visual sensors have been proposed in recent years, aiming at executing real-time vision tasks with reduced energy budget [5]–[8]. The most straightforward way to do it is to embed sensing and analog processing into the same chip and to downscale the supply voltage. However, more careful energy management and processing strategies can be adopted to further reduce the overall power consumption. Among others, one of the most critical is to reduce the chip activity at the IOs. This can be done through efficient data encoding and compressing and, whenever it is possible, by forcing the sensor or part of the sensor into low-power sleep mode. Although in-sensor analog computation is extremely energy-efficient, it lacks programmability, which makes such approach not suitable to extract middle- and high-level features.

In this work, we present an Ultra-Low-Power (ULP) embedded vision system, which is at the same time energy efficient and highly flexible. The proposed smart visual sensor is composed of an ULP 128x64 contrast-based imager [9], consuming $10.6\mu W$ at 10fps and typical pixel activity measured during benchmarking tests, and a fully programmable 4-cores ULP platform (PULP [3]), featuring a power consumption of $2.9mW$ at the frequency of $82MHz$, and supply voltage V_{DD} of 0.56V. In this architecture, we combine the analog imager processing, aimed to produce visually relevant events, with the near-threshold, event-triggered and fully programmable parallel digital processing, to extract high-level features. By exploiting embedded analog processing, the imager internally performs pixel-level contrast extraction, binarization and temporal frame-differencing, and produces address-event coded information, here referred as *pixel events*. The PULP platform processes the arrays of visual events produced by the imager to extract high-level information.

The processing of visual data is event-driven, inspired by neuromorphic computing [10]. According to the event-driven model, the digital computation occurs only when relevant events are detected by the imager. Hence, if no moving objects appear in the scene, the processing unit can be kept in deep sleep mode. Moreover, only the relevant events are transferred to the MCU's main memory, resulting in significant saving of communication bandwidth and energy with respect to traditional frame-based sensors. We demonstrate that, in our case-study vision application, the overall processing energy cost per frame can be reduced by a factor 383x with respect to an embedded vision system which employs Commercial Off-The-Shelf (COTS) ULP components and a traditional frame-based computational approach consisting of frame segmentation and connected components extraction.

Summarizing, the main contributions of this work are:

1) The definition and design of the system architecture, the interfaces between sensor and digital processing engine and the power management strategy.

2) The implementation of event-driven algorithms for moving objects detection, which are optimized to exploit the PULP parallel execution features.
3) The detailed quantification of the energy efficiency improvements of the proposed system with respect to a traditional frame-based vision system based on a ULP COTS sensor and MCU.

In the following, Section II gives an overview of the related works and in Section III our embedded vision system is presented. The event-driven approach and its implementation and optimization are respectively discussed in sections IV and V. In Section VI we report the performance evaluation of the proposed system within the considered application framework and Section VII concludes the paper.

II. Related Work

Traditionally, smart cameras with in-node digital visual processing capabilities show a power consumption of several hundreds of mW [11]–[13]. Such systems feature VGA or higher resolution cameras and can perform complex local vision tasks (e.g. object detection and tracking). In [13], authors report a power consumption of $514.8mW$ for imager activity, not including MCU, external memory and transmission module. Low-level strategies to reduce power on visual sensor node are presented in [14]: authors perform acquisition down-sampling at hardware-level on the MCU and, after estimating the target location, only the correspondent region-of-interest is transferred from the imager to the MCU's main memory. Alternative solutions have also been presented. *MeshEye* [15] is a heterogeneous camera mote, which hosts up to 8 low power and low resolution cameras (ADNS-3060 optical mouse sensors 30x30 pixel 6-bit grayscale) and a VGA sensor. The core unit is an Atmel MCU, which incorporates a 32 bit RISC architecture ARM7TDMI Thumb processor, clocked at 55 MHz. To save energy, object motion detection is performed on the low resolution cameras, which, if necessary, wake up the VGA camera. *Cyclops* [16] is equipped with an ATmega128 8-bit RISC microcontroller clocked at 7.3 MHz and a CMOS image sensor delivering RGB images at CIF resolution. The authors report a consumption of $42mW$ during image capturing, $23mW$ for MCU internal operation and less than $1mW$ in sleep state. Because of its limited computational power, Cyclops was employed for first-level light video processing in a multi-tier camera network [17].

Technological advances in microelectronics have allowed the integration of pixel-wise features extraction circuits on the same die of the image sensor. These smarter vision chips, such as [4], [5], [18], are able to perform analog early vision processing and generate post-processed data. A 64x64 imager with on-chip clustering-based processing has been presented in [19]. The vision chip embeds an event-driven algorithm while consuming $0.4mW$ at $100fps$. It can localize up to three regions in the scene corresponding to moving objects. The sensor achieves high energy efficiency but it lacks flexibility, which instead can be achieved by employing a digital signal processor for data post-processing. On this side, Wi-FLIP [20] couples a smart vision chip FLIP-Q, which incorporates pixel-level processing elements with a commercial processing platform [21], featuring a 32-bit PXA271 XScale processor. Regarding such architecture, the low-power imager consumption ($5.6mW$) represents the 5.2% of the system power budget, while the rest is due to the digital processor. In [22], authors present a smart camera incorporating the SCAMP-3 vision chip [23], an FPGA and an ARM Cortex-M3 MCU. By exploiting early vision processing, a power consumption of few tens of mW is measured for object tracking and counting.

In an effort to radically increase energy efficiency, bio-inspired vision sensors attempt to mimic the extremely efficient visual system of living organisms [24], [25]. Silicon retinas have been implemented as array of pixels where each pixel handles its own information and dispatches data by means of an event-based asynchronous protocol, called address-event representation (AER) [26]. A 120x120 event-based imager for low-power mobile applications has been shown in [27], consuming $500\mu W$ in normal mode and $250\mu W$ in stand-by mode, which is an order of magnitude higher than the power consumption of our imager. Event-driven data processing has been introduced to extract relevant features from visual signal early-processed by the retina, without any frame timing reference [10]. This is in contrast with frame-based processing, where all frames must be entirely processed pixel-by-pixel, even if they do not contain any significant data, therefore requiring high computational power and resources. An event-driven approach for clustering events associated with moving objects is presented in [28]. Computation on event-based sensor data becomes extensive considering the random order, either spatial and temporal, of the high amount of events detected (a fast moving ball generates several thousands of events per second [29]). Despite several of previous works exploit desktop PCs for event-based data processing [29], [30], an interesting embedded event-based system *eDVS* was presented in [31] and used within robotic applications [32]. Such system showed a power consumption as low as 23mW. To the best of our knowledge, power management opportunities emerging from the coupling of an ultra-low power processing platform with an event-based sensors have not been fully explored in the open literature. One shortcoming of purely asynchronous sensors is that any processing unit coupled to such a sensor should be able to handle the peak rate of asynchronous events, which is generally orders of magnitude larger than the average. On the contrary, our imager preserves the frame timing and outputs the addresses of the significant (asserted) pixel-events, while respecting the raster-scan order. Hence, with respect to event based systems, the digital processing occurs at fixed-rate on limited spatially ordered sets of events, allowing to better fit memory and computational constraints of ultra-low-power embedded systems and to exploit very energy-efficient power management strategies.

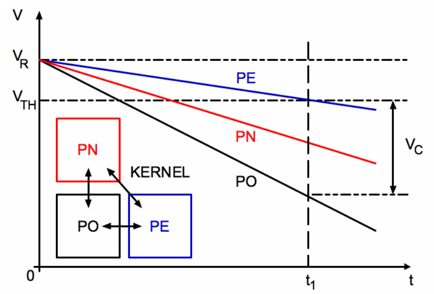
In the proposed architecture, we combine the ultra-high-efficiency of the ULP imager with near-threshold digital parallel computing. Event-driven processing of imager data is performed on the fully-flexible platform and optimized for parallel runtime.

III. System Overview

In this section we provide an overview of the system, describing the main components. The block diagram of the system is reported in Fig. 3.

A. Imager

The ultra low power imager [9] features analog pixel-level spatial-contrast extraction and binarization. During the sensor exposure time, each pixel estimates the spatial contrast (V_C) over a 3 pixel kernel (Fig. 1) and binarizes the value against a threshold (V_T).



$$V_C(t_1) = (V_R - V_{TH}) \cdot (\frac{I_{PO} - I_{PE}}{I_{PE}})$$

Fig. 1. Basic operating principle of contrast extraction.

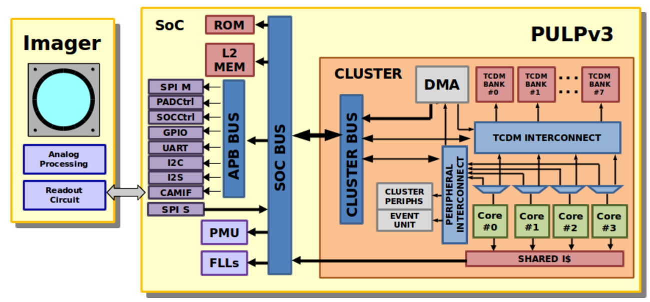


Fig. 3. Block diagram of the proposed smart camera node architecture. The smart vision chip coupled with the PULPv3 multicores processing platform.

The pixel detecting a contrast larger than the threshold is asserted (active pixel). The basic pixel operations are shown in Fig. 2. The pixel has a 1-bit memory that can be used to store the binary contrast of the past frame to be compared with the current one, in order to implement motion detection through frame difference.

During the imager readout, only the active pixels are delivered off-chip, thus reducing the amount of data and the sensor power consumption as well. Instead of dispatching a bitmap, only the address of each asserted pixel is provided. Therefore, in motion detection mode, the imager streams an array of pixels, referred also as events, when moving objects appear in the field of view.

B. PULPv3 SoC

PULP (Parallel processing Ultra-Low Power platform) is a multi-core platform that exploits parallel, near-threshold operation to satisfy the computational requirements of a wide range of applications requiring near-sensors processing, constrained by power budgets of few mW [3]. The third embodiment of the PULP platform (PULPv3) is described in the following. The compute engine is a cluster with 4 cores. The ISA and the core micro-architecture feature optimizations for energy efficient digital signal processing, supporting zero-overhead hardware loops with L0 I-buffer, load and store operations embedding pointer arithmetic, SIMD vector instructions and power management instructions [33]. The cluster features 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. The cores share 4Kb of instruction cache with support for broadcast to exploit the Single Instruction Multiple Data (SIMD) behaviour of several signal processing algorithms, further increasing energy efficiency. Off-cluster (64kB) L2 memory and peripheral access latency is managed by a tightly coupled DMA optimized for low power.

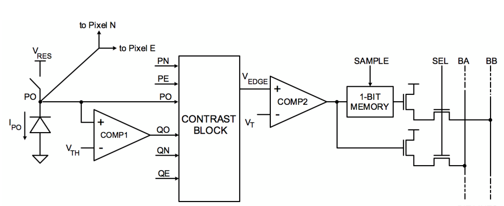


Fig. 2. Block diagram of the pixel.

Several peripherals are available on the SoC, including SPI interfaces with streaming support, I2C, I2S, a camera interface, GPIOs, and a bootup ROM. In the context of this work, we assume the ULP imager connected to the PULP SoC through the camera interface. To provide high energy efficiency across a wide range of workloads, the PULP cluster and the rest of the SoC are in different power and clock domains. Fine-grained tuning of the SoC and cluster frequencies is achieved through two FLLs (Frequency-Locked Loops). A power management unit (PMU) automatically manages transitions of the cluster between the active and deep-sleep states. The cluster can be put in deep-sleep state with a write operation on a memory mapped control register, while the SoC goes in a low-power wait-for-event mode. After going in sleep mode, the cluster remains in idle mode until a configurable event is triggered. Events can be issued by all IO peripherals, or by a timer.

IV. Event-Driven Moving Object Detection

An event-driven approach for object localization and tracking using an event-based sensor was originally presented in [28]: authors refer to "cluster" as base computation element to identify a group of pixel with circular bounding box and the clustering of events is conducted according to distance criterion. Drawing inspiration from this previous work, we develop an event-driven clustering approach to be applied to the array of N asserted pixel addresses, also named events, dispatched from our imager at frame time k. The developed algorithm sequentially scans the ordered array to group events into rectangular-shaped clusters, named blobs. For each frame k, the detected blobs are stored in a list $\mathcal{L}(k)$. Any blob B of $\mathcal{L}(k)$ is described by the following features: the center of mass $\overline{x}_c(B,k)$, the rectangular-shaped bounding box, expressed by its boundary coordinates $[\mathrm{x}_{min}(B,k), \mathrm{y}_{min}(B,k)]$ and $[\mathrm{x}_{max}(B,k), \mathrm{y}_{max}(B,k)]$, and the number of pixels $W(B,k)$ within the blob (i.e. its area). Similarly to [28], we define a *seek-region*, according to the following constraints:

$$S_{\mathrm{x}}(B,k) = \min\{R_{MAX}, \frac{\mathrm{x}_{max}(B,k) - \mathrm{x}_{min}(B,k)}{2} + \delta\}$$

$$S_{\mathrm{y}}(B,k) = \min\{R_{MAX}, \frac{\mathrm{y}_{max}(B,k) - \mathrm{y}_{min}(B,k)}{2} + \delta\}$$

where R_{MAX} and δ are parameters of the algorithm and the minimum condition assures that the region is kept within certain limits. In these equations, S_x and S_y define the distances of the seek-region limits from the center of mass.

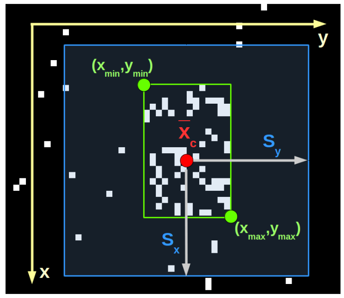


Fig. 4. Blob detected on an imager scene. The bounding box is depicted in green while the seek region is represented in blue.

It is introduced to manage the blob formation, as explained below. An example of seek region is reported in Fig. 4, along with the bounding box and the center of mass of a detected blob.

At frame time $k+1$ the imager outputs a set of asserted pixels $\{\overline{p}_{i,k+1} := \overline{p}(i, k+1)\}_{i=0,..N}$. The following tasks are executed on the array of events to extract the blob features:

I Blob Formation

Ia For each event $\overline{p}_{i,k+1}$, the system selects, if any, a blob B of $\mathcal{L}(k)$ if $\overline{p}_{i,k+1}$ is within the seek-region and such that the Euclidean distance between the center of mass $\overline{x}_c(B, k)$, and $\overline{p}_{i,k+1}$ is the smallest one.

Ib Pixel not assigned during the previous step, are sequentially scanned to form new blobs. Events are processed respecting the readout raster-scan order. The first event not assigned at point (Ia) seeds a new blob. From the second onward, the procedure tries to assign the pixel to one of the new formed blobs according to the same criteria of previous step. Centers of mass, bounding boxes and seek regions need to be re-computed after pixel assignment. If an event has not been assigned to any new blob, it will seed a new one.

II Blob Filtering
The list $\mathcal{L}(k+1)$ is filtered by removing blobs with a too small area. This task is implemented by two steps, that filter respectively blobs formed based on previous frame information (point (Ia)) and blobs formed from scratch (point (Ib)).

III Blob Merging
Blobs whose bounding boxes have a large intersection are merged together.

V. Optimization of the Event-Driven Algorithm for ULP Parallel Processing

This section describes the optimization of the algorithm described in Section IV on the PULP platform. For embedded vision systems, algorithm design and code optimization are typically addressed to deal with smart camera limited resources [1]. In this work, to speed-up the execution time and consequently reduce the processor's energy consumption, two main actions are performed:

- Exploitation of PULPv3 instruction set extensions [33] to improve the execution performance.
- Algorithm flow optimization to allow parallel workload distribution over multi-core platform (Fig. 5).

The proposed event-driven approach works on events coordinates. Therefore, algorithm implementation greatly benefits from the PULPv3 SIMD vector operations. By using such processor extension, the spatial coordinates of either pixel events and blob features (center of mass, boundary coordinates of bounding box, seek-region distances) can be efficiently handled as 2x16 bit vectors.

To exploit the full computational power of PULPv3 4-cores cluster, the computational load is distributed among the available cores, by means of thread-level parallelism. Our parallel strategy is validated against a video dataset described in section VI-A. Typically, the Blob Formation phase of points (Ia)-(Ib) of the algorithm represents the heaviest computational section of the entire procedure. The operations at point (Ia) are highly parallelizable and only few write operations are coded in a critical section to handle mutually exclusive access to the blob list $\mathcal{L}(k)$, that is instantiated as a shared variable. If a pixel cannot be assigned during this step, it will be processed for the formation of new blobs (point (Ib)). To preserve the raster-scan order processing, the described operation has to be executed sequentially on the remaining events by a single core. This fact represents a critical bottleneck for the parallel runtime. For instance, if no blobs are found in a frame, the blob formation phase in the successive frame needs to be entirely performed on a single core.

On average, 95.5% of the algorithm execution time is spent on the blob formation phase, but only 36.2% of such time is spent on the parallelizable block (point Ia). The filtering (points (II)) of new blobs is executed on multiple cores, by partitioning the number of blobs among all cores. Instead, the filtering of blobs formed based on previous information is performed on a single core because of the low number of items (refer to Table II). The final merge operation (points (III)) gathers all the remaining blobs and try to merge them, therefore the code is sequentially executed. Applying Amdahl's law reveals a maximum speed-up of 1.4x, which appears to be very limited with respect to maximum 4x.

To overcome this limitation, we modify the algorithm as illustrated in fig. 5b. For each frame k, the event array $\overline{p}_{i,k}$ is partitioned in 4 array subsets, each of them managed by a separate thread. Each thread handles the assignment of events with respect to blob features computed in the previous frame (point (Ia)) and the formation of blobs from scratch (point

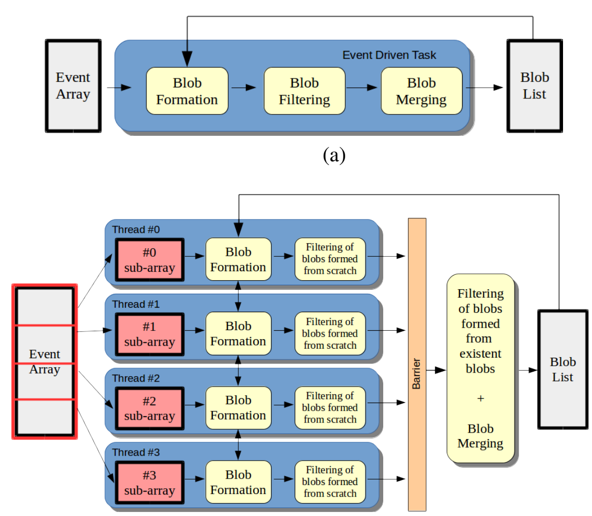


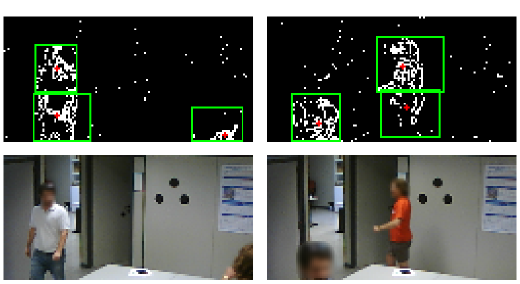
Fig. 5. The event-driven algorithm flow is reported in (a) while in (b) the performed optimization to target PULP 4-cores platform is depicted

(Ib)). The former operation requires a locked shared memory space where every thread pushes the assigned events. Each blob memory space has its own lock to reduce contention. The formation of blobs from scratch, which represents the major issue for parallelization of the first algorithm implementation, is now split into sub-task to be concurrently handled by multiple cores: every thread builds a private list of new blobs only processing the events of the sub-array while respecting the raster-scan order. The new detected blobs can be then independently filtered before the merging phase. When each thread runs on one of the four cores of the platform, a synchronization point is placed after filtering the private blob list. Once all threads reach the barrier, a final task, executed on a single core, filters the blobs formed with respect to previous frame features, gathers all new filtered blobs and tries to merge together all the extracted blobs. Tests on the video dataset reveals that 94% of the execution time is spent on the parallalizable block, which turns into a theoretical maximum speed-up of 3.39x according to Amdahl's law.

VI. Experimental Results

A. Experimental Setup

We measure the performance of the event-driven system on a real-life application that consists in monitoring an indoor space. The integration time is set to $100\mathrm{msec}$ (corresponding to $10\mathrm{fps}$), which is considered suitable for monitoring applications. We collect six video sequences, each composed by 340 frames. Benchmark videos, along with ground truth images and post-processing information, are publicly available at [34]. Some examples are shown in Figure 6. After collecting camera data, we evaluate the performance of our parallel implementation in comparison with a single core implementation. In particular, we run the single-core implementation on both an ARM Cortex M4 based MCU and on a single core of the PULPv3 cluster. For each video, the blob detection starts by assuming that no blobs are previously identified. The PULPv3 cycle-accurate FPGA emulator is used to gather runtime statistics about the video processing. In Table I we report some information about the benchmarking videos. The videos show one (single-object) or two (multi-objects) people moving in an indoor environment, with different speed and direction. 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 number of events per frame and consequently the imager bandwidth and the required memory footprint depend on the context activity. The values of these features (denoted by "Avg Pixel", "Imager BW" and "Memory", respectively), averaged over the number of frames per video, are reported in Table I. In addition, we report the number of average and peak MOPS[1] ("Avg MOPS" and "Peak MOPS"), needed to execute the algorithm.



(a) (b)

Fig. 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 bottom.

TABLE I
Experimental Dataset Features

Benchmark Video	*vid0*	*vid1*	*vid2*	*vid3*	*vid4*	*vid5*
Avg Pixels (N/ frame)	99.6	138.6	118.5	176.1	104.3	138.2
Memory [Bytes]	199	277	237	352	209	276
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

B. Performance Evaluation

The event-driven local processing remarkably reduces the amount of output data from the smart visual sensor. In Table II, we report the number of detected blobs on each video sample averaged over the number of frames ("Avg Detected Blobs"). Since each blob descriptor occupies 32bytes of memory space, we calculate the embedded system bandwidth to send out the detected blob information ("System BW") and the correspondent bandwidth reduction with respect to that of the imager reported in Table I. On average, the bandwidth is reduced by 91x.

To quantify the accuracy and precision of the event-driven local processing, we manually crop the bounding boxes of ground truth objects. Then we compute the following measures:

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

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;

$$precision = \frac{n_{target}}{n_{target} + n_{fp}} \tag{2}$$

$$recall = \frac{n_{target}}{n_{target} + n_{fn}} \tag{3}$$

where n_{target}, n_{fp} and n_{fn} denote respectively the number of marked ground-truth objects, false positives and false negatives. On our video dataset we obtain an accuracy of 0.70 and 0.71 respectively for the event-driven blob detection algorithm and its optimized version. The precision achieved is 0.95 for the baseline low-parallelism algorithm, while the algorithm optimized for parallelism achieves 0.93. Recall is above 0.98 on both cases. Hence, the even-based approach is effective and its optimizations for increased efficiency do not compromise accuracy, precision and recall.

In Table III, the proposed event-driven system is compared with a traditional frame-based embedded vision system that

[1]Equivalent OpenRISC operations.

TABLE II
Blob detection results

Benchmark Video	*vid0*	*vid1*	*vid2*	*vid3*	*vid4*	*vid5*
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 III
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

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 our sensor thanks 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, we report in the table 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. We note that in the frame-based approach several filters have to be applied to each frame (frame difference, binarization, dilation and erosion) before extracting the connected components.

Figure 7 summarizes the execution time of the event-driven filter on different platforms, with and without the algorithm optimization of Section V. We report the number of clock cycles normalized with respect to those required to run the video benchmarks on an ARM-Cortex M4 core (exploiting only 32-bit arithmetic instructions). On PULP single core, the average execution time is reduced by 4% and a further reduction of 25% is achieved by exploiting its ISA extensions (indicated with label *HW Ext* in the figure). With the first attempt of parallelization, intrinsically limited by Amdahl's law, a speed-up of 1.27 and a consequent execution time reduction of 21% are obtained. The 4-thread algorithm version (labelled with *OPT* in the figure) presents a total speed-up of 2.5x instead of the ideal value 3.39x. The lower speed-up is due to the unbalancing of threads concurrently running on the 4 cores (48% of the overhead) and to the accesses to critical section, parallelization overhead and contention in L1 memory. By running the optimized version on the 4-cores cluster, a further execution time reduction of 60% is obtained with respect to that of not-optimized parallel implementation.

C. System Power Estimation

In this section we analyse the power consumption of the presented event-driven system along with a comparison with a COTS embedded vision system, which employs a traditional frame-based computational model.

TABLE IV
PULP PERFORMANCE AND POWER ESTIMATIONS

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

PULPv3 cluster power model is applied for the analysis of processing energy costs. Fine-grained tuning of cluster voltage allows the selection of the most energy-efficient operating point. The cluster power densities, along with the maximum frequencies, are illustrated in table IV 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 [35]. 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 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, we assume an active power consumption of $1mW$ for the SoC, that includes L2, bus, clock and supply voltage generation and IOs.

To estimate the processing energy cost per frame within our vision application, we model the event-driven execution 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 in deep sleep mode. To estimate the average energy consumption within the frame period we consider power consumptions of both active mode and deep sleep mode. The number of cycles required to execute the task, along with a given frequency, determines the time period of active mode. For application with very low duty cycle, the deep sleep power becomes relevant, or even dominant. On PULPv3 platform, by considering the leakage

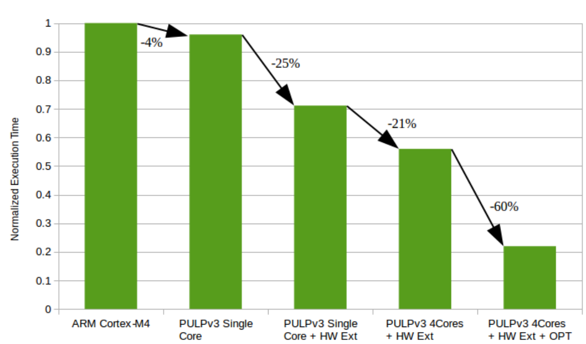


Fig. 7. Comparison of software execution time (clock cycles) for different algorithm implementations. Values are normalized with respect to the execution time of the not-optimized algorithm running on an ARM Cortex-M4 core.

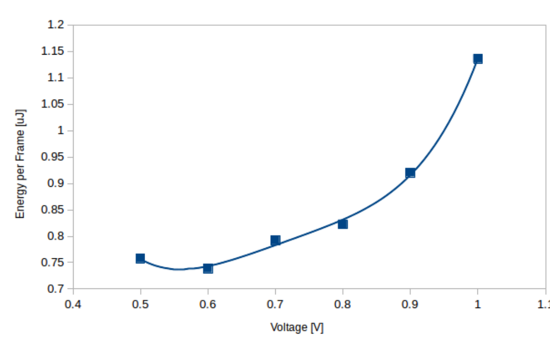


Fig. 8. PULPv3 processing energy cost per frame on different operating points

TABLE V
SYSTEM ENERGY COSTS ESTIMATION AND COMPARISON

Scenario	Frame Based	Event Driven			
	Frame-Based Camera [38] + STM32 [37]	Event-Based Sensor [27] + STM32 [37]	Our Imager [9] + STM32 [37]	Our Imager [9] + Apollo [36]	Our Imager [9] + PULPv3 [3]
Average pixels per Frame	8192	130	130	130	130
Imager Energy [μJ/frame]	**62.2**	**28.4**	**1.06**	**1.06**	**1.06**
Processing Clock Frequency [MHz]	26	26	26	24	82
Processing Active Power [mW]	8.6	8.6	8.6	2.7	2.9
Duty Cycle	72.1%	1.21%	1.21%	1.31%	0.11%
Processing Energy [μJ/frame]	**623.7**	**10.82**	**10.82**	**3.67**	**0.73**
System Energy [μJ/frame]	**685.9**	**39.22**	**11.88**	**4.73**	**1.79**

power of SoC, L2 memory and IO pads required to implement the protocol with the imager, the deep sleep power amounts to $4.2\mu W$. In this estimation, we consider a 32 kHz clock to drive the SoC always-on region. Figure 8 illustrates the processing energy cost per frame on several energy-efficient operating points. 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.

In table V the overall system energy cost per frame is compared with other ultra low power solutions. Power consumption measurements related to the image sensor are measured on silicon samples [9]. We scale down the power consumption according to sensor typical activity observed during the execution of our benchmarks. For comparison purpose, the memory requirements of our vision application impose the selection of alternative processing platforms with at least 32kBs memory RAM. We refer to the *Ambiq Apollo* [36], which features an ARM Cortex-M4F core and up to 64kBs RAM, and to the off-the-shelf *STM32L476* [37], an ULP MCU with an ARM Cortex M4 core and 128kB SRAM. Among MCUs, *Ambiq Apollo* achieves the lowest power in sleep mode, as 0.33μW, 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 COTS MCUs. It 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 behaviour. For fair comparison, we take into account the processor clock cycles needed to run the optimized algorithm on an ARM Cortex M4 core.

Moreover, in Table V our event driven system is compared with a 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 [38] and an event-based camera [27]. Both sensor power consumption are linearly scaled to match the resolution and the frame-rate of our imager. In the event-based imagining system, we define a time window as long as our frame period and we assume to retrieve the same data from the event-based sensor with respect to our imager within this time window. As a consequence of these optimistic assumptions, the system is able to exploit an efficient race-to-halt run model and can be kept in deep sleep mode for long time. We remark however that this optimized power management strategy has not been presented yet, to the best of our knowledge, in the open literature.

[2]As deep sleep mode, we refer to *STM32L476*'s Stop2 Mode

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 near-threshold operation, 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 9 reports the system energy reduction. By coupling PULPv3 with the imager, our proposed node architecture achieves an energy cost per frame of 1.79μJ, 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 event-based imaging system based on ultra-low-power asynchronous camera. 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 average power consumption of $17.9\mu W$.

VII. CONCLUSION

In this work, we presented a smart camera sensor architecture, targeting ultra-low-power vision applications, and its usage within a case-study of moving objects detection. The system contains a contrast-based imager and an ultra-low-power parallel processing platform (PULP). Besides the individual low power consumption, the imager features continuous analog processing to produce significant vision events. The

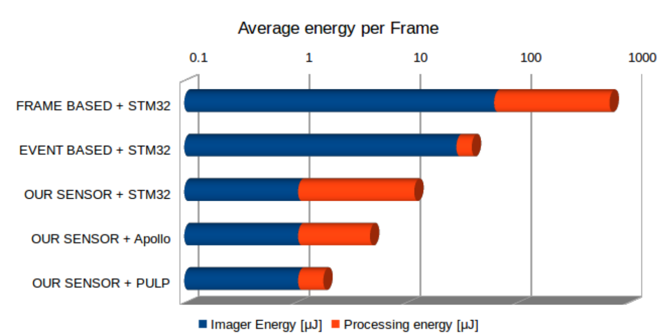


Fig. 9. System Energy Cost Comparison

pixel address coding readout behaviour allows to reduce the data bandwidth by 31x, compared to an imager that continuously sends full pixel frames. High-level features are extracted by means of event-driven processing on a fully programmable multi-core platform. We describe the implementation and optimization strategies to target multi-cores embedded systems. Compared to most common approaches, based on image processing from traditional frame-based cameras, the event-driven operation applied to an ultra-low-power MCU platform, allows to reduce by more than 57.7x the system energy processing per frame. We exploit the energy-efficient parallel near-threshold processing on PULPv3 platform to further reduce by 6.64x the energy cost, achieving a system energy consumption of 1.79μJ per frame. The proposed architecture leads to an overall energy boost of 383x and 21.8x with respect to, respectively, a traditional frame-based visual system and an event-base imaging system based on asynchronous visual sensor. Powering the system with a coin cell battery will result in about 248 weeks of battery life.

References

[1] A. N. Belbachir, *Smart cameras*. Springer, 2010.

[2] T. Torfs, T. Sterken, S. Brebels, J. Santana, R. van den Hoven, V. Spiering, N. Bertsch, D. Trapani, and D. Zonta, "Low power wireless sensor network for building monitoring," *Sensors J., IEEE*, vol. 13, no. 3, pp. 909–915, 2013.

[3] D. Rossi, A. Pullini, M. Gautschi, I. Loi, F. K. Gurkaynak, P. Flatresse, and L. Benini, "A -1.8v to 0.9v body bias, 60 gops/w 4-core cluster in low-power 28nm utbb fd-soi technology," in *SOI-3D-Subthreshold Microelectronics Technology Unified Conf. (S3S), 2015 IEEE*, 2015, pp. 1–3.

[4] S. Chen, W. Tang, X. Zhang, and E. Culurciello, "A 64 x 64 pixels uwb wireless temporal-difference digital image sensor," *Very Large Scale Integration (VLSI) Systems, IEEE Trans. on*, vol. 20, no. 12, pp. 2232–2240, 2012.

[5] J. Choi, S. Park, J. Cho, and E. Yoon, "A 3.4μw cmos image sensor with embedded feature-extraction algorithm for motion-triggered object-of-interest imaging," in *Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), 2013 IEEE Int.* IEEE, 2013, pp. 478–479.

[6] T. Ohmaru, T. Nakagawa, S. Maeda, Y. Okamoto, M. Kozuma, S. Yoneda, H. Inoue, Y. Kurokawa, T. Ikeda, Y. Ieda *et al.*, "6.5 25.3μw at 60fps 240$\times$ 160-pixel vision sensor for motion capturing with in-pixel non-volatile analog memory using crystalline oxide semiconductor fet," in *Solid-State Circuits Conf.-(ISSCC), 2015 IEEE Int.* IEEE, 2015, pp. 1–3.

[7] G. Kim, M. Barangi, Z. Foo, N. Pinckney, S. Bang, D. Blaauw, and D. Sylvester, "A 467nw cmos visual motion sensor with temporal averaging and pixel aggregation," in *Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), 2013 IEEE Int.* IEEE, 2013, pp. 480–481.

[8] A. Berkovich, M. Lecca, L. Gasparini, P. A. Abshire, and M. Gottardi, "A 30 μw 30 fps 110$\times$ 110 pixels vision sensor embedding local binary patterns," *Solid-State Circuits, IEEE J. of*, vol. 50, no. 9, pp. 2138–2148, 2015.

[9] M. Gottardi, N. Massari, and S. A. Jawed, "A 100 w 128 64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *Solid-State Circuits, IEEE J. of*, vol. 44, no. 5, pp. 1582–1592, 2009.

[10] T. Delbruck, "Frame-free dynamic digital vision," in *Proc. of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, 2008, pp. 21–26.

[11] P. Chen, K. Hong, N. Naikal, S. S. Sastry, D. Tygar, P. Yan, A. Y. Yang, L.-C. Chang, L. Lin, S. Wang *et al.*, "A low-bandwidth camera sensor platform with applications in smart camera networks," *ACM Trans. on Sensor Networks (TOSN)*, vol. 9, no. 2, p. 21, 2013.

[12] A. Rowe, C. Rosenberg, and I. Nourbakhsh, "Cmucam website, http://www.cmucam.org/projects/cmucam5."

[13] T. Winkler and B. Rinner, "Secure embedded visual sensing in end-user applications with trusteye.m4," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth Int. Conf. on*. IEEE, 2015, pp. 1–6.

[14] M. Casares, S. Velipasalar, P. Santinelli, R. Cucchiara, and A. Prati, "Energy-efficient feedback tracking on embedded smart cameras by hardware-level optimization," in *Distributed Smart Cameras (ICDSC), 2011 5th ACM/IEEE Int. Conf. on*. IEEE, 2011, pp. 1–6.

[15] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Proc. of the 6th int. conf. on Information processing in sensor networks*. ACM, 2007, pp. 360–369.

[16] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "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.

[17] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *Proc. of the 13th annual ACM int. conf. on Multimedia*. ACM, 2005, pp. 229–238.

[18] J. Fernández-Berni, R. Carmona-Galán, R. del Río, R. Kleihorst, W. Philips, and Á. Rodríguez-Vázquez, "Focal-plane sensing-processing: A power-efficient approach for the implementation of privacy-aware networked visual sensors," *Sensors*, vol. 14, no. 8, pp. 15 203–15 226, 2014.

[19] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang, "A 64 64 cmos image sensor with on-chip moving object detection and localization," *Circuits and Systems for Video Technology, IEEE Trans. on*, vol. 22, no. 4, pp. 581–588, 2012.

[20] J. Fernández-Berni, R. Carmona-Galán, G. Linán-Cembrano, Á. Zarándy, and Á. Rodríguez-Vázquez, "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.

[21] L. Nachman, "Intel corporation research santa clara. ca. new tinyos platforms panel: imote2," *The Second International TinyOS Technology Exchange*, 2005.

[22] S. J. Carey, D. R. Barr, and P. Dudek, "Low power high-performance smart camera system based on scamp vision sensor," *J. of Systems Architecture*, vol. 59, no. 10, pp. 889–899, 2013.

[23] P. Dudek and S. Carey, "General-purpose 128$\times$ 128 simd processor array with integrated image sensor," *Electronics Letters*, vol. 42, no. 12, pp. 678–679, 2006.

[24] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *Solid-State Circuits, IEEE J. of*, vol. 43, no. 2, pp. 566–576, 2008.

[25] J. Lenero-Bardallo, P. Hafliger, R. Carmona Galan, and A. Rodriguez-Vazquez, "A bio-inspired vision sensor with dual operation and readout modes," *Sensors J., IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[26] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: bioinspired cameras with spiking output," *Proc. of the IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.

[27] R. Berner, P. Lichtsteiner, T. Delbruck, J.-H. Kim, K. Lee, J. Lee, K.-H. Park, T. Kim, and H.-Y. Ryu, "Dynamic vision sensor for low power applications," in *Consumer Electronics (ISCE 2014), The 18th IEEE Int. Symposium on*. IEEE, 2014, pp. 1–2.

[28] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schön, B. Kohn, and H. Garn, "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.

[29] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, 2007, pp. 845–848.

[30] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor," *Neuromorphic Engineering Systems and Applications*, p. 16, 2015.

[31] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An embedded aer dynamic vision sensor for low-latency pole balancing," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 780–785.

[32] G. R. Müller and J. 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.

[33] M. Gautschi, A. Traber, A. Pullini, L. Benini, M. Scandale, A. Di Federico, M. Beretta, and G. Agosta, "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.

[34] Video dataset - URL: https://e3da.fbk.eu/projects/energy-efficient-smart-vision-systems.

[35] P. Flatresse, B. Giraud, J. Noel, B. Pelloux-Prayer, F. Giner, D. Arora, F. Arnaud, N. Planes, J. Le Coz, O. Thomas *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.

[36] Ambiq Apollo website. URL http://ambiqmicro.com/low-power-microcontroller.

[37] STMicroelectronics. STM32L476 Datasheet.

[38] Aptina. MT9V011 VGA CMOS Active-Pixel Digital Image Sensor.

This work
was supported in part by the ICYSOC RTD Project evaluated by the Swiss National Science Foundation through Nano-Tera.ch with
Swiss Confederation financing, and in part by Union EuroCPS
H2020-ICT-2014 under Grant 644090.