Bottom-up Performance Analysis of Focal-plane Mixed-signal Hardware for Viola-Jones Early Vision Tasks

J. Fernández-Berni*, R. Carmona-Galán, R. del Río and Á. Rodríguez-Vázquez

Institute of Microelectronics of Seville (IMSE-CNM),
Consejo Superior de Investigaciones Científicas y Universidad de Sevilla,
C/ Américo Vespucio s/n, 41092, Seville, Spain (e-mail: berni@imse-cnm.csic.es).

SUMMARY

Focal-plane mixed-signal arrays have traditionally been designed according to the general claim that moderate accuracy in processing is affordable. The performance of their circuitry has been analyzed in these terms without a comprehensive study of the ultimate consequences of such moderate accuracy. In this paper, for the first time to the best of our knowledge, we do carry out this study. We move expectable performance of mixed-signal image processing hardware directly into the vision algorithm making use of it. This permits to close a wider design loop, enabling a more aggressive design of this kind of hardware provided that the algorithm, at the highest level —semantic interpretation of the scene—, can afford it. Thus, we present a thorough analysis of the non-idealities associated with the implementation of a QVGA array tailored for the distinctive characteristics of the Viola-Jones processing framework. The resulting deviation models are then introduced in the processing flow of this framework provided by the OpenCV library. We have found, contrary to what could be expected, that these deviations do not necessarily degrade the performance of the Viola-Jones algorithm. They could be even beneficial for certain high-level specifications. Additionally, we demonstrate the architectural advantages of our approach: exploitation of focal-plane distributed memory and ultra-low-power operation.

KEY WORDS: Viola-Jones framework, focal-plane sensing-processing, mixed-signal circuitry, integral images, Haar-like features, OpenCV library

1. INTRODUCTION

Back in the early 90's, CMOS technology enabled a functional coexistence of photo-sensors and processors on a chip, thus allowing the incorporation of intelligence into solid-state imaging

*Correspondence to: berni@imse-cnm.csic.es

Contract/grant sponsor: Ministerio de Economía y Competitividad (Spain); Office of Naval Research (USA); CDTI (Spain); contract/grant number: TEC2012-38921-C02-01, co-funded by the European Regional Development Fund, and IPT-2011-1625-430000; grant N000141110312; IPC-20111009, co-funded by the European Regional Development Fund

devices [1]. The degree of embedded intelligence has ranged widely from simply control and image correction, typically found in most current commercial image sensors, to more elaborated tasks backing up higher cognitive vision capabilities. Significant research activity is focused on the latter. Different strategies have been proposed to increase and improve the smart features of image sensors [2]. *Focal-plane sensing-processing* [3] constitutes the best approach in terms of exploitation and adaptation to the particular characteristics of early vision [4]. On the one hand, the information to be handled at this processing stage —each and every pixel resulting from the raw readings of the sensors— is massive. On the other hand, the computational flow is very uniform. The same calculations are repeatedly carried out on every pixel. More interestingly, the outcome for each individual pixel does not usually depend on the outcome for the rest. Consequently, while an enormous amount of data must certainly be processed, regular massively parallel operation can still be applied. Focal-plane sensor-processor chips make the most of these characteristics by operating in Single Instruction Multiple Data (SIMD) mode [5] featuring concurrent processing and distributed memory. Numerous low-level image processing primitives have been successfully implemented following this scheme: convolution filtering [6,7], programmable blurring [8], spatial [9] and temporal [10, 11] contrast extraction, background subtraction [12], image compression [13] or high dynamic range imaging [14] among others. Even academic [15,16] and commercial [17] general-purpose vision systems based on focal-plane processing have been reported.

Focal-plane processing architectures can also benefit from the possibility of including *analog circuitry*. When compared to their digital counterpart, analog circuits can reach higher performance in terms of speed, area and power consumption, but at the cost of low, moderate at most, accuracy. It has been generally claimed that most vision algorithms can perform properly under these conditions [18]. There is however a lack of in-depth analysis in the literature about the ultimate impact of such moderate accuracy on high-level vision processes delivering the semantic interpretation of a scene. This analysis is specially relevant when it comes to application-specific chips. For them, the study of the ultimate consequences of decisions made at electrical or even physical design level can lead to solutions much closer to the optimal for the targeted functionality.

All in all, the contribution of this paper is twofold: i) we address, for the first time to the best of our knowledge, the design of focal-plane mixed-signal circuitry implementing early vision tasks required by the Viola-Jones processing framework [19,20]; ii) we carry out, by making use of exhaustive simulations and the OpenCV library [21], a comprehensive bottom-up analysis of how the non-idealities of the physical realization affect the final outcome of the algorithm underlying the mentioned framework. This analysis enables a hardware-software co-design loop that boosts the confidence in achieving the targeted functionality from the resulting array. In addition to other capabilities, this array provides support to the Viola-Jones processing flow at two low-level stages. First, it is be able to deliver the integral and square integral images at different scales. Alternatively, it can compute the sum of pixels and squared pixels at any possible rectangular area of the image, significantly easing the extraction of Haar-like features.

The paper is organized as follows: the Viola-Jones framework is outlined in Section 2, highlighting those tasks presenting a feasible focal-plane implementation. The methodology and circuitry proposed to address such implementation are described in Section 3. Simulation results are shown in Section 4. Finally, we analyze in Section 5 how these results affect the outcome of the Viola-Jones face detection baseline algorithm provided by the OpenCV library.

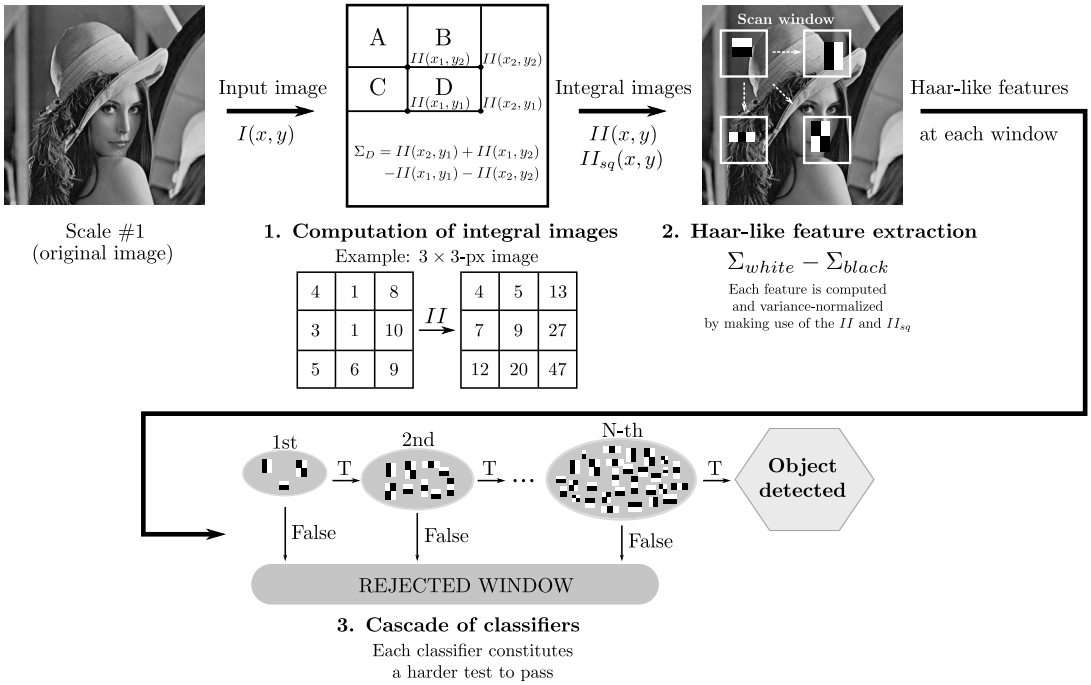

Figure 1. Simplified scheme of the Viola-Jones processing flow.

2. VIOLA-JONES PROCESSING FRAMEWORK

The Viola-Jones framework constitutes one of the best approaches reported to achieve real-time object recognition. It is based on the extraction of very simple features across the image —the so-called *Haar-like features*— which are subsequently analyzed by a cascade of classifiers of progressive complexity. These classifiers are previously trained according to the object to be detected, adapting their internal thresholds when successive training images are passed through. A basic scheme of the Viola-Jones processing flow is depicted in Fig. 1. Despite its simplicity, this framework still requires a considerable amount of computational and memory resources. During the last few years, numerous efforts have been focused on exploiting the increasing memory and logic capabilities available in FPGAs [22–24] as well as the highly parallel computation structure of GPUs [25, 26]. When it comes to low-power embedded systems, additional constraints must be introduced on the image resolution [27] or the type of processor operations [28] in order to obtain at least moderate frame rates.

From the point of view of focal-plane processing, we are interested in the early stages of the flow represented in Fig. 1. From now on, we will be considering the most usual operation mode for the Viola-Jones framework [28]. In this mode, the original image is scaled down until reaching a prescribed minimal dimension. The processing flow is repeated for each scaled image. Note that the raw material feeding the cascade of classifiers consists of Haar-like features. These features derive from the Haar wavelets [29] and encode differences in average intensities

between rectangular regions. Their mathematical formulation is extremely simple. For a certain feature F_k, we have that:

$$F_k = \sum_i \sum_j W_{ij} - \sum_p \sum_q B_{pq} \tag{1}$$

where W_{ij} represents the pixel values within the white rectangle(s) and B_{pq} the pixel values for the black rectangle(s). White and black are mere indicators of the area considered, with independence of their pixel values. In practice, we are simply comparing the DC component of the rectangles involved since the sum of the pixels is proportional to their mean value.

The large amount of resources to be allocated for the algorithm comes from the correspondingly large number of Haar-like features to compute. As an example, the Viola-Jones face detection algorithm provided by the OpenCV library requires 22 classifiers including 2135 features in total. Of course, most of the windows scanned across the image are rejected at the first —and simpler— classifiers of the cascade on not containing the targeted object. This avoids a great deal of useless calculations. But still there will be windows in which all the features will have to be checked. In order to alleviate the computational and memory requirements from this processing stage, an intermediate image representation is used, the so-called *integral image*. This intermediate representation is defined as:

$$II(x,y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} I(x',y') \tag{2}$$

where $I(x,y)$ represents the input image. That is, each pixel composing $II(x,y)$ is given by the sum of all the pixels above and to the left of the corresponding pixel at the input image. Two fundamental advantages support the inclusion of this pre-processing stage. First of all, only four pixels adequately extracted from the integral image permit to compute the sum of any rectangular region of the input image. Consider four points as in Fig. 1, (x_1,y_1), (x_2,y_1), (x_1,y_2) and (x_2,y_2), with $x_1 < x_2$ and $y_1 < y_2$, defining a rectangle across the input image. The sum of pixels within this region can be expressed as:

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} I(x,y) = II(x_2,y_1) + II(x_1,y_2) - II(x_1,y_1) - II(x_2,y_2) \tag{3}$$

The second advantage is that the integral image can be computed in one pass over the input image by making use of the following pair of recurrences:

$$\begin{cases} r(x,y) = r(x,y-1) + I(x,y) \\ II(x,y) = II(x-1,y) + r(x,y) \end{cases} \tag{4}$$

with $r(x,0) = 0$ and $II(0,y) = 0$. This single-pass computation enables a fast operation on the part of a microprocessor.

In addition to the integral image, the Viola-Jones processing flow also requires the calculation of the *square integral image*, defined as:

$$II_{sq}(x,y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} I^2(x',y') \tag{5}$$

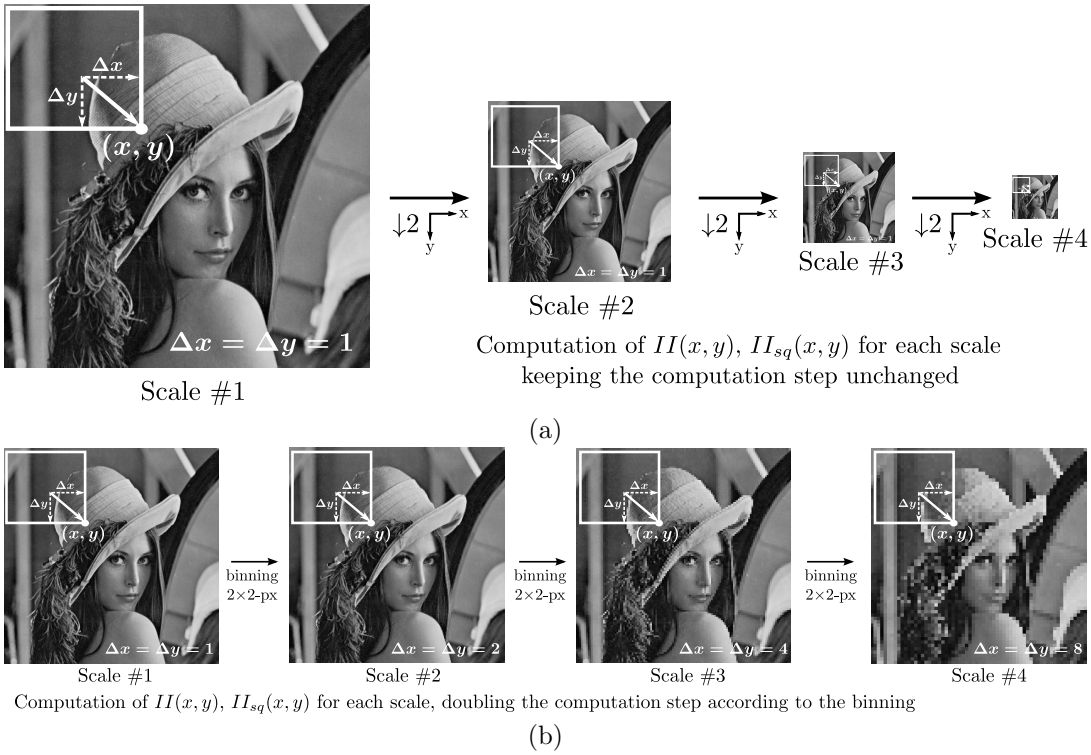

Figure 2. The original raw image and successively downsampled versions of it (a) require the computation of their corresponding integral images. In practice, this is equivalent to make use of the original raw image and its successive versions obtained by pixel binning (b).

This extra intermediate representation allows, in conjunction with $II(x,y)$, the variance normalization of the Haar-like features. All the windows used for classifier training are variance-normalized in order to minimize the effect of different lighting and contrast conditions. Correspondingly, the features extracted from the input image must also be variance-normalized. Taking into account that the variance of the generic rectangle previously defined for Eq. (3) can be expressed as:

$$\sigma^2 = \frac{1}{MN}\sum_{x=x_1}^{x_2}\sum_{y=y_1}^{y_2} I^2(x,y) - \left[\frac{1}{MN}\sum_{x=x_1}^{x_2}\sum_{y=y_1}^{y_2} I(x,y)\right]^2 \tag{6}$$

where M and N denotes the horizontal and vertical dimensions in pixels of the rectangle, and considering the counterpart of Eq. (3) for $II_{sq}(x,y)$, we can re-write Eq. (6) as:

$$\begin{aligned}\sigma^2 \quad = \quad & \tfrac{1}{MN}\left[II_{sq}(x_2,y_1) + II_{sq}(x_1,y_2) \;\; -II_{sq}(x_1,y_1) - II_{sq}(x_2,y_2)\right] \\ & -\left\{\tfrac{1}{MN}\left[II(x_2,y_1) + II(x_1,y_2) \;\; -II(x_1,y_1) - II(x_2,y_2)\right]\right\}^2\end{aligned} \tag{7}$$

which shows how both integral images work together to achieve the variance normalization.

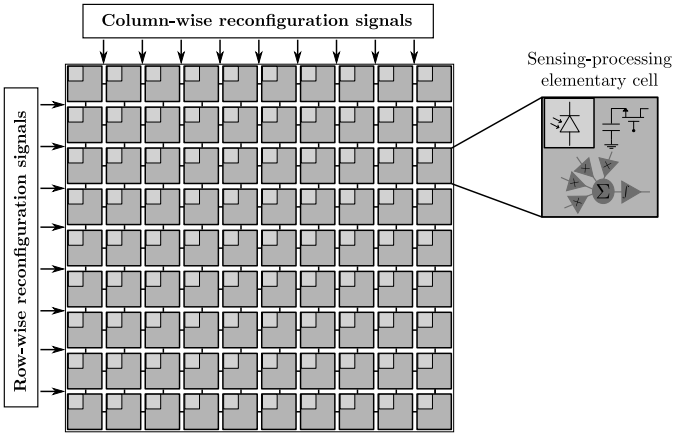

Figure 3. Focal-plane sensing-processing scheme proposed for the computation of integral images.

In the next section, we will propose a processing scheme devised for the focal-plane computation of $II(x, y)$ and $II_{sq}(x, y)$. This computation, that constitutes the lowest-level task for the Viola-Jones framework, can clearly benefit from the concurrent operation and distributed memory provided by focal-plane architectures. We will thoroughly describe the mixed-signal circuitry implementing integral image computation and analyze its performance. We will also demonstrate that the high degree of reconfigurability of the proposed scheme would even permit to go further by skipping integral image computation and enabling a direct extraction of Haar-like features through the concurrent computation of rectangular image area sums. However, the exploitation of this strategy requires a substantial modification of the algorithm at software level that falls beyond the scope of the paper.

3. IMPLEMENTATION PROPOSAL

Our objective is the implementation of reconfigurable focal-plane circuitry delivering integral images at different scales, as depicted in Fig. 2(a). For the sake of hardware simplicity, this is equivalent in practice to make use of the original image and its successive versions obtained by pixel binning, as represented in Fig. 2(b). For each scale, the pixels are correspondingly merged through averaging and the computation step along the x and y axis is doubled. In order to reach the aforementioned objective, we propose a general scheme like that of Fig. 3. A focal-plane array of 4-connected sensing-processing elementary cells provides the computational and memory resources required. These cells, whose interconnection can be reconfigured by means of peripheral circuitry, operate in a massively parallel way. They will work concurrently and jointly according to the corresponding instruction. Note however that such parallelism cannot be applied to obtain all the pixels of an integral image at the same time. Assuming a $W \times H$ array, it would mean to hold $W \times H$ copies of the top-left pixel of the original image since this pixel is needed for the computation of each and every pixel of the integral image. Likewise,

a progressively reduced number of pixel copies along the original image would also have to be held. Instead, we propose the exploitation of concurrency and distributed memory during a sequential processing stage. The circuitry to achieve this is shown in Fig. 4. It has been designed for the standard UMC 0.18μm 1P6M CMOS process. After photointegration, the pixel is represented by the voltage V_{ij}. This voltage is repeatedly copied into $V_{S_{ij}}$ by enabling the analog buffer through the control signal CP_EN and then squared at $V_{SQ_{ij}}$ in order to respectively support the computation of the integral and square integral images. We re-use the squarer reported in [31] because of its simplicity and successful experimental verification. The sum of pixels and squared pixels is carried out through charge redistribution enabled by the switches controlled by $\mathrm{EN}_{S_{i,i+1}}$, $\mathrm{EN}_{S_{j,j+1}}$, $\overline{\mathrm{EN}_{SQ_{i,i+1}}}$ and $\overline{\mathrm{EN}_{SQ_{j,j+1}}}$. These signals are set by peripheral circuitry according to the scale and current pixel location of the integral images being calculated, as will be explained shortly. A timing diagram for two consecutive stages of copy, squaring and charge redistribution for scale #1 is depicted in Fig. 5. Once redistributions take place in parallel at $V_{S_{ij}}$ and $V_{SQ_{ij}}$, these voltages constitute new pixels of the targeted integral images. Charge redistribution can really be described as a diffusion process defined, for example for $V_{S_{ij}}$, as:

$$R_S C_S \frac{dV_{S_{ij}}}{dt} = -4V_{S_{ij}} + V_{S_{i+1,j}} + V_{S_{i-1,j}} + V_{S_{i,j+1}} + V_{S_{i,j-1}} \tag{8}$$

where R_S is the equivalent resistance of the switches and C_S is the capacitance holding $V_{S_{ij}}$. Eq. (8) is for an inside cell like that of Fig. 4 featuring full connectivity. Cells at the edges are connected to fewer than four neighbors. Unlike the implementation presented in [31], we are not now interested in transient states of this diffusion process but in the steady state. And we want to attain it as fast as possible. Consequently, the switches must be as wide as area restrictions allow, thereby reducing their resistance. The steady state of a diffusion process like that of Eq. (8) is characterized by a uniform distribution of voltages across the group of cells involved. Every voltage reaches the same value, that coincides with the average of the initial voltages at the cells. It is this average what encodes the sum required by the integral images.

In order to better visualize how the charge redistribution is configured, a simplified scheme of the proposed array is shown in Fig. 6. It can be seen that the cells can be grouped column-wise and row-wise through the corresponding control signals. Each pixel of the integral images is related to a stage of copy, squaring and charge redistribution. After these three steps, the array must be re-arranged for the next pixel. As an example, the computation of the first row of the integral images at scale #1 requires to disable all row connections between cells and then progressively enable column connections. Thus, if we focus on $\mathrm{EN}_{S_{i,i+1}}$, the column interconnection pattern '0000...0' leads to $II(1,1)$, '1000...0' to $II(2,1)$, '1100...0' to $II(3,1)$ and so on. Applying ones' complement to these patterns, those of $\overline{\mathrm{EN}_{SQ_{i,i+1}}}$ for $II_{sq}(1,1)$, $II_{sq}(2,1)$, $II_{sq}(3,1)$, etc. are respectively obtained. For further scales, a more complex redistribution arrangement is needed. To explain this, let us describe peripheral circuitry capable of providing $\mathrm{EN}_{S_{i,i+1}}$ and $\overline{\mathrm{EN}_{SQ_{i,i+1}}}$ —exactly the same is used for $\mathrm{EN}_{S_{j,j+1}}$ and $\overline{\mathrm{EN}_{SQ_{j,j+1}}}$. It is depicted in Fig. 7. We basically require a shift register like in [31]. This makes reconfiguration for scale #1 very simple and also ease further processing capabilities [32]. But the point is how to deal efficiently with successive scales. Keep in mind that we first need pixel binning and then reconfigurable charge redistribution over the resulting image. To speed up these two tasks, we incorporate the possibility of disabling the shift register and setting interconnection patterns in parallel through the signals denoted as $SC\#$. These signals are

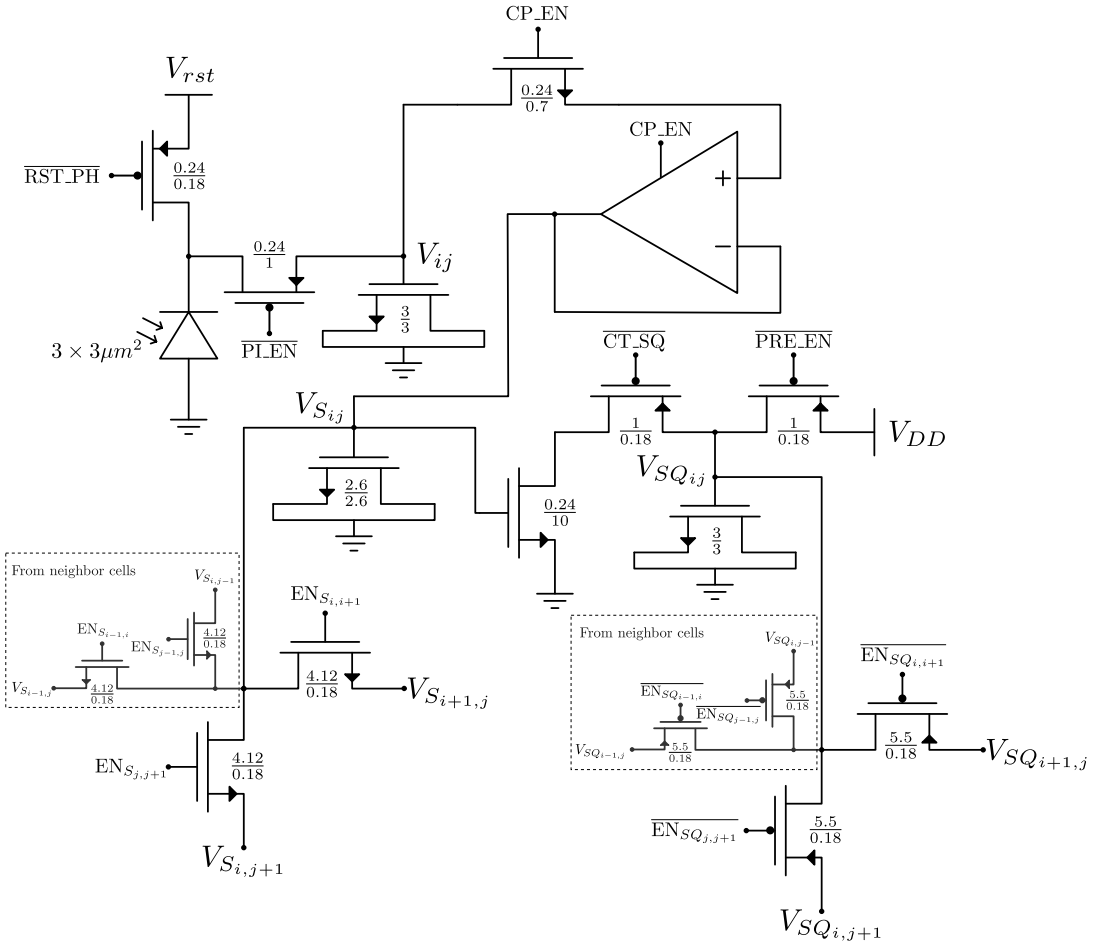

Figure 4. Proposed circuitry for integral image computation at the elementary sensing-processing cell.

distributed along the peripheral cells as illustrated in Fig. 8. For scale #1, binning is not necessary. Our starting point is therefore an all-0's bit string for rows and columns. For scale #2, binning is achieved by switching the signal $SC2$, distributed as indicated in Fig. 8, from logic '0' to '1'. In so doing, we merge voltages $V_{S_{ij}}$ and $V_{SQ_{ij}}$ within 2×2-px blocks. By switching also the signal $SC3$ to '1', again as distributed in the figure, the merging process would affect blocks of 4×4-px. Likewise, $SC4$ is associated with 8×8-px blocks and $SC5$ with 16×16-px blocks. A single signal therefore permits to re-arrange the array for the next scale. The final step is to perform charge redistribution between the macro-pixels thus generated. This can be done by loading the adequate interconnection patterns, similarly to scale #1. Taking scale #2 as an example, and assuming again the computation of the first row, the column interconnection pattern '1000...0' would lead to $II(1,1)$, '111000000..0' to $II(2,1)$,

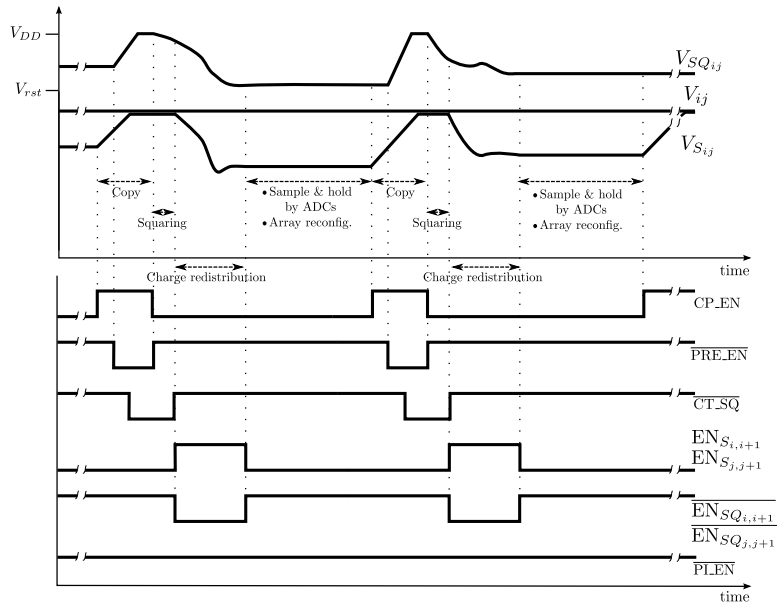

Figure 5. Timing diagram of two consecutive stages of copy, squaring and charge redistribution at an elementary cell for scale #1.

'111110000..0' to $II(3,1)$, etc. These patterns mean to double the computation step along the x axis with respect to that of scale #1, accordingly to the macro-pixel dimensions. It is this enormous flexibility for focal-plane reconfiguration what endows the array with the additional asset of computing the sum of pixels and squared pixels at multiple rectangular areas in parallel, as required for the direct extraction of Haar-like features.

As a final comment, notice that, in order to read out and convert every pixel of the integral images, we must simply connect $V_{S_{1,1}}$ and $V_{SQ_{1,1}}$ to respective analog-to-digital converters located at the periphery of the array. These voltages will always contain the targeted calculation for each pixel at each scale, according to the definition of integral image and the proposed hardware implementation based on charge redistribution.

4. ERROR CHARACTERIZATION

We have built up a 320×240-px array by making use of the circuitry depicted in Fig. 4 plus additional transistors and another photodiode in order to exploit the array reconfigurability for block-wise intra-frame HDR imaging [30]. The resulting layout is shown in Fig. 9. Post-layout simulation results will endow our analysis with very high reliability concerning expectable performance from the array. The array is surrounded by peripheral circuitry providing the functionalities already described.

In order to evaluate how the non-idealities of the proposed circuitry impact the targeted computation of integral images, we have thoroughly analyzed the elementary cell after RC

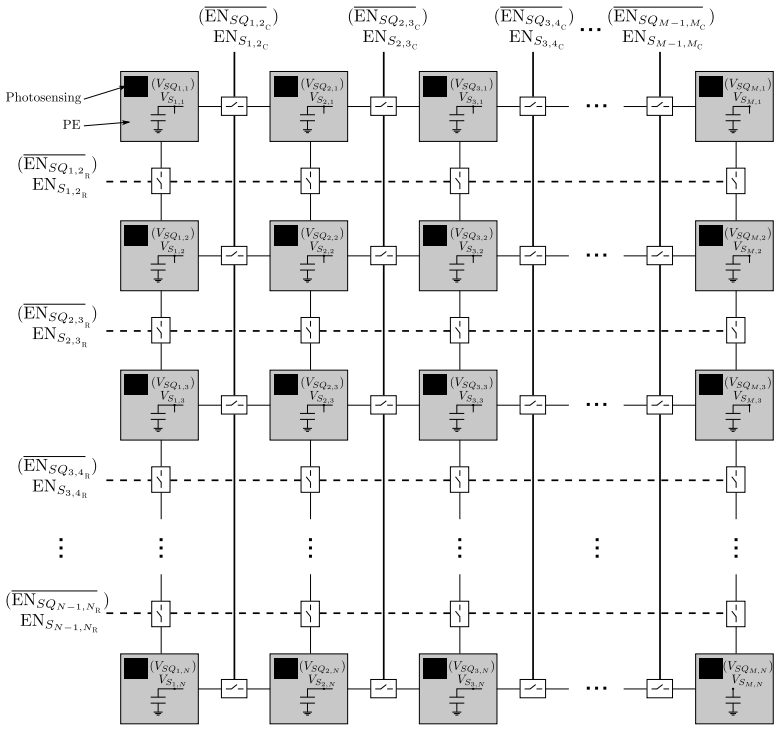

Figure 6. Simplified scheme of how the charge redistribution can be reconfigured in the proposed focal-plane array.

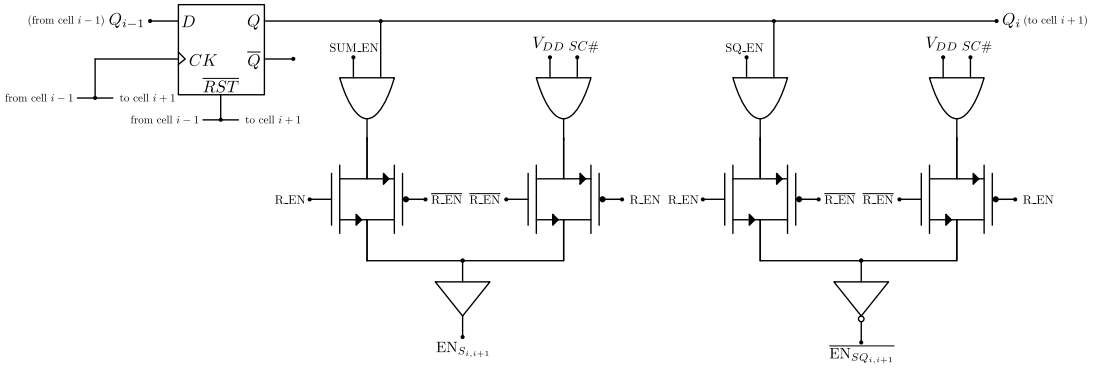

Figure 7. Peripheral cell per column connection. The same cell is used per row connection in order to provide $\mathrm{EN}_{S_{j,j+1}}$ and $\overline{\mathrm{EN}_{SQ_{j,j+1}}}$.

parasitic extraction. Specifically, five primary sources of deviation have been studied, namely: charge injection on pixel voltage, mismatch between sensing capacitances, error in pixel copy,

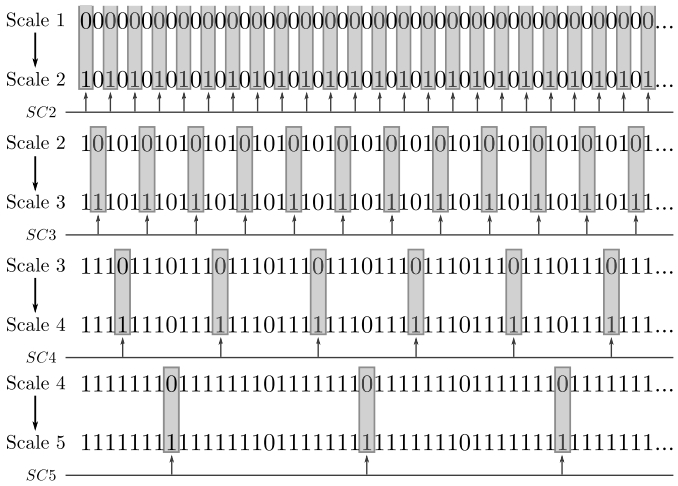

Figure 8. Distribution of the signals $SC\#$ in order to set five scales.

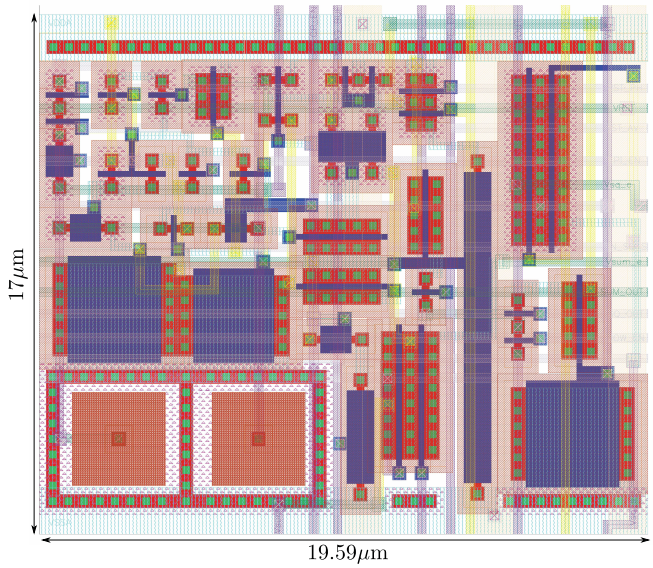

Figure 9. Layout of the elementary cell including the circuitry depicted in Fig. 4 plus additional transistors and another photodiode for block-wise intra-frame HDR imaging.

error in squaring and error in charge redistribution. The worst-case scenario under global process parameter variations at the different technology corners and local mismatch for each corner is always considered. MOSFET models (HSPICE level 49 BSIM3V3.2) provided by the manufacturer are used. Let us start by charge injection on pixel voltage. This signal-dependent

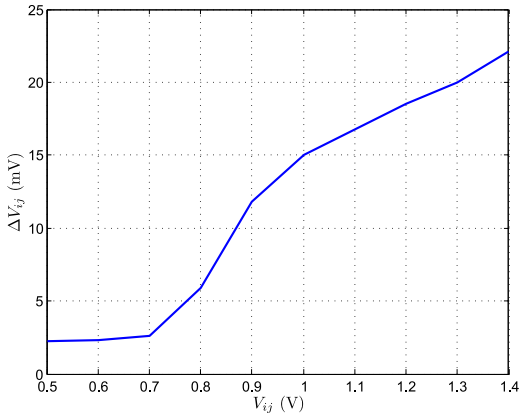

Figure 10. Signal-dependent positive offset undergone by V_{ij} due to charge injection after photointegration.

deviation takes place when photointegration finishes by switching OFF the pMOS switch acting as a transmission gate. In order to characterize it, we have simulated different photodischarge operations ending at a number of pixel voltages along the signal range. The photodischarge is stopped by switching from ON to OFF the pMOS transistor, recording the change undergone by the pixel at that moment. The worst case occurs for the 'FF' corner, where 30 Monte Carlo mismatch simulations lead to positive average offsets from 2.3mV for a pixel voltage of 0.5V up to 22.1mV for a pixel voltage of 1.4V, as depicted in Fig. 10. The standard deviation is negligible, ranging from 0.03mV for 0.5V to 0.2mV for 1.4V.

Mismatch between sensing capacitances is the main parameter of the fixed-pattern noise that we can reliably estimate since the manufacturer does not provide information about the mismatch and photo-response of the photodiodes employed. Ideally, the pixel value is inversely proportional to the sensing capacitance:

$$V_{ij} \propto \frac{I_{ph_{ij}} T_{int}}{C} \tag{9}$$

where $I_{ph_{ij}}$ is the average photocurrent generated at pixel (i, j) during the integration period T_{int} and C is the value of the MOS capacitance holding V_{ij}. Due to mismatch, we will have a certain deviation on this capacitance that can be expressed as:

$$V_{ij} \propto \frac{I_{ph_{ij}} T_{int}}{C} \frac{1}{1 + \delta_{ij}} \tag{10}$$

In order to determine the magnitude of δ_{ij}, we connected the MOS capacitor —and all its associated parasitics— to an ideal grounded resistor of 1kΩ. This value of resistance is arbitrary. Its only function is to provide a reference for comparison with an ideal RC circuit. The reset voltage, 1.4V in our case, was then set as the initial condition of the resulting RC circuit and 30 Monte Carlo transient simulations were carried out at all the corners of the technology. For each simulation, the dynamics along the signal range was compared with that of the corresponding ideal RC circuit obtained by least square fitting. The first point we must remark is the good

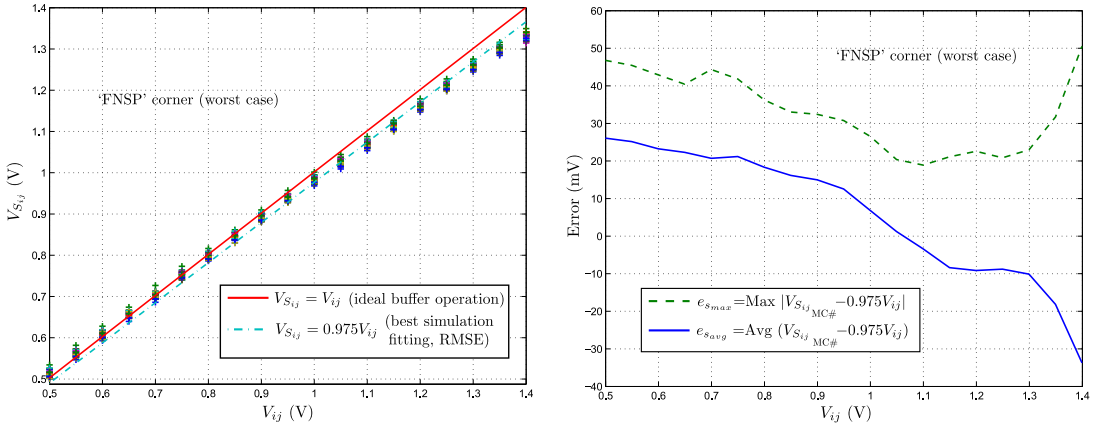

Figure 11. Least square fitting of 30 Monte-Carlo simulations of the buffer operation (a) and extracted signal-dependent error parameters (b). The 'FNSP' corner (worst case) is considered in both plots.

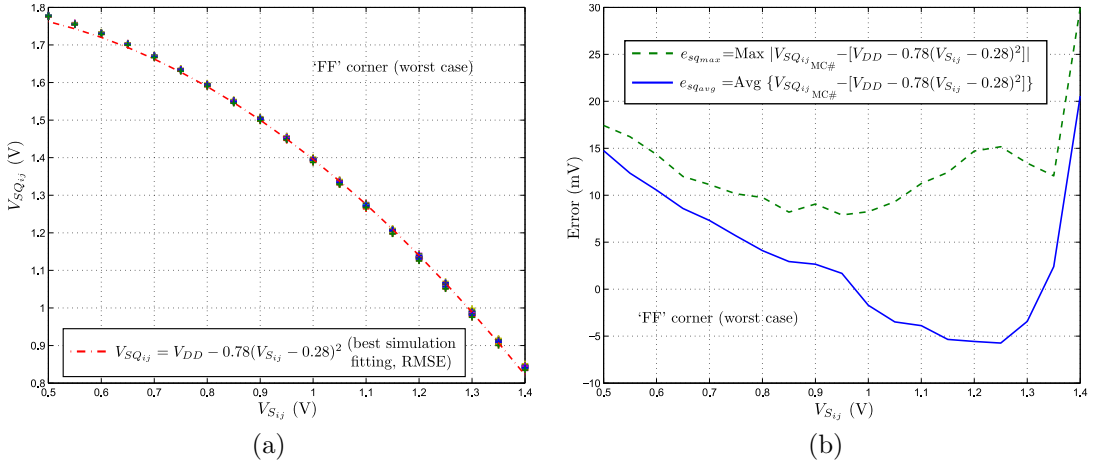

Figure 12. Least square fitting of 30 Monte-Carlo simulations of the squaring operation (a) and extracted signal-dependent error parameters (b). The 'FF' corner (worst case) is considered in both plots.

linearity of the MOS capacitors. For the worst case, the 'SS' corner, the average root-mean-square error (RMSE) between the simulated and the ideal dynamics was 1.96%. In this corner, the average capacitance was 76.662fF with a standard deviation of 0.012fF. This value could seem very small, but it makes sense when taking into account the size of the design grid of the process, 10nm, the gate oxide thickness and the dimensions of the capacitor. It is also consistent with the very low fixed pattern noise —0.4% of voltage swing— experimentally measured from a smart imager manufactured in the same technology reported in [33]. We thus model δ_{ij} as a random variable picking up values across the array with normal distribution:

$$\delta_{\mathbf{ij}} = N(\mu, \sigma^2) = N[0, (0.012/76.662)^2] \tag{11}$$

A much greater deviation is associated with the pixel copy. The ideal input-output voltage transfer of the analog buffer along the signal range is compared in Fig. 11(a) with the outcome of 30 Monte-Carlo transient simulations performed at the 'FNSP' corner —worst case in terms of RMSE for this operation. The maximum time interval required by the buffer to achieve the pixel copy is 150ns. A suitable deviation model to accommodate this input-output transfer function is given by:

$$\mathbf{V_{S_{ij}}} = k\mathbf{V_{ij}} + \mathbf{e_{offset}} \tag{12}$$

where we can distinguish a gain error (k) and an offset error of statistical nature ($\mathbf{e_{offset}}$). The gain error term can be considered a constant according to the least square fitting of the simulation results presented in Fig. 11(a), leading to $k = 0.975$. The offset term can be mathematically described by means of two signal-dependent parameters quantified in Fig. 11(b): an average offset $e_{s_{avg}}$ and a maximum offset, in absolute value, $e_{s_{max}}$. The parameter $e_{s_{avg}}$ describes the average offset due to local mismatch variations whereas the parameter $e_{s_{max}}$ corresponds to the absolute value of the maximum deviation caused by mismatching. The offset term $\mathbf{e_{offset}}$ can thus be defined as:

$$\mathbf{e_{offset}} = e_{s_{avg}} + \Delta\mathbf{e_{offset}} \tag{13}$$

that is, the offset is equal to the average offset plus a certain deviation added to this mean value due to mismatching. Finally, to determine this deviation, we set a pessimistic interval of variation defined by:

$$\Delta\mathbf{e_{offset}} = (e_{s_{max}} - |e_{s_{avg}}|)\mathbf{U}(-\mathbf{1}, \mathbf{1}) \tag{14}$$

where the use of absolute value is mandatory in order to account for negative average offsets. This equation means that there can be a positive or negative maximum deviation on the average offset with equal probability than any other intermediate deviation. If a Gaussian distribution were used instead, the maximum deviations would be much less probable. This is the reason behind making use of a uniform distribution. The final error model, according to Eq. (12), is therefore defined by:

$$\mathbf{V_{S_{ij}}} = 0.975\mathbf{V_{ij}} + e_{s_{avg}} + (e_{s_{max}} - |e_{s_{avg}}|)\mathbf{U}(-\mathbf{1}, \mathbf{1}) \tag{15}$$

The pixel squaring operation takes place faster than the pixel copy. Only 25ns are required once the MOS capacitor is precharged to V_{DD} —this precharge is realized during the pixel copy, as illustrated in Fig. 5. The resulting voltage $V_{SQ_{ij}}$ can be expressed as:

	4×4-px	**8×8-px**	**16×16-px**	**32×32-px**	**64×64-px**	**128×128-px**
$\lvert\Delta V_{S_{1,1}}\rvert$	0.57%	0.33%	0.19%	0.17%	0.16%	0.08%
$\lvert\Delta V_{SQ_{1,1}}\rvert$	0.31%	0.16%	0.25%	0.18%	0.14%	0.12%

Table I. Maximum error, in terms of percentage of signal range, for the operation of charge redistribution when dismissing charge injection and clock feedthrough.

$$V_{SQ_{ij}} = V_{DD} - k_1(V_{S_{ij}} - k_2)^2 \tag{16}$$

where the parameters k_1 and k_2 demand external calibration since they are strongly dependent on global process variations, as explained next. In order to characterize this operation, we set different voltages along the signal range at $V_{S_{ij}}$. For each voltage, the corresponding value of $V_{SQ_{ij}}$ is obtained, enabling the least square fitting of Eq. (16). For 30 Monte-Carlo mismatch simulations at each process corner, the worst case occurs for the 'FF' corner, as depicted in Fig. 12(a). The deviation parameters $e_{sq_{max}}$ and $e_{sq_{avg}}$, calculated similarly to $e_{s_{max}}$ and $e_{s_{avg}}$, are also extracted in Fig. 12(b). The statistical deviation model of pixel squaring at this corner can thus be built as:

$$\begin{aligned} \mathbf{V_{SQ_{ij}}} \quad = \quad & V_{DD} - 0.78(\mathbf{V_{S_{ij}}} - 0.28)^2 + e_{sq_{avg}} \\ & +(e_{sq_{max}} - |e_{sq_{avg}}|)\mathbf{U}(-\mathbf{1}, \mathbf{1}) \end{aligned} \tag{17}$$

where again $e_{sq_{max}}$ and $e_{sq_{avg}}$ depends on the input voltage, in this case $\mathbf{V_{S_{ij}}}$. Note that Eq. (17) defines the worst case in terms of local variations for particular values of k_1 and k_2 related to a specific corner. Unfortunately, these parameters feature a significant variability across the technology corners: k_1 varies between $0.57\text{V}^{-}1$ and $0.78\text{V}^{-}1$ whereas k_2 ranges from 0.28V to 0.41V. Thus, for calibration of different array instances, a similar procedure to the characterization just described can be followed. The voltage V_{rst} can externally be swept along the signal range while making use of the control signals $\overline{\text{RST\_AV}}$ and $\overline{\text{PI\_EN}}$ to set different voltages at $V_{S_{ij}}$. The corresponding voltage $V_{SQ_{ij}}$ can thus be achieved and read out for external fitting, leading to particular values of k_1 and k_2.

The last error to be analyzed is related to charge redistribution. There are three sources of deviation for this operation: the mismatch between the different transistors involved, the charge injection generated by the switches when they are set ON and OFF to start and stop the redistribution and finally, the clock feedthrough associated with the switching of the control signals. The deviation caused by mismatch is negligible. We have initialized the MOS capacitors, both those used for computing the integral image and the ones providing the square integral image, according to a scaled version of the Lena image for different sizes of block. The dynamics of each RC network is then released until the steady state (99% of the expected value) is reached. The maximum error, in terms of percentage of signal range, for the top-left pixel after 30 Monte-Carlo simulations with typical transistor models for each size of block is represented in Table I. These results hardly change across the rest of corners. The key point in these simulations is that the transistors acting as switches were already ON before releasing the dynamics and therefore their channel was already created. However, this is not realistic since we have to make use of their gate control signals during the operation of the array,

	4×4-px	**8×8-px**	**16×16-px**	**32×32-px**	**64×64-px**	**128×128-px**
$\Delta V_{S_{1,1}}$	+3.83%	+5.14%	+6.17%	+6.52%	+6.69%	+6.79%
σ	0.02%	0.03%	0.01%	0.01%	0.02%	0.02%
$\Delta V_{SQ_{1,1}}$	-3.77%	-5.68%	-6.36%	-6.74%	-6.97%	-6.98%
σ	0.01%	0.01%	0.02%	0.02%	0.01%	0.02%

Table II. Average error along with the corresponding standard deviation for the operation of charge redistribution at the 'FF' corner, taking into account now channel charge injection and clock feedthrough.

thereby creating and destroying their channel. When considering these signals, channel charge injection and clock feedthrough become major sources of deviation. The average error along with the corresponding standard deviation, again expressed as percentage of signal range, after 30 Monte-Carlo simulations for the 'FF' corner —worst case— in these conditions is shown in Table II[†]. Note that, unlike in Table I, we are not considering the absolute value of the error in Table II. Due to the dominance of charge injection as the primary source of deviation, and the different type of switches used —nMOS for the integral image and pMOS for the square integral image—, the error always features a positive offset for $V_{S_{1,1}}$ and a negative offset for $V_{SQ_{1,1}}$. We will therefore model this deviation as an offset depending on block size, added to the final value of the integral and square integral sums. The numerical values presented in Table II are taken as reference for any other possible images being processed. It must be noticed that charge redistribution can be a limiting factor when it comes to pushing the frame rate up. The reason is the time interval required to reach the steady state when the whole array is involved in the redistribution. While for small blocks this interval does not go beyond few nanoseconds, a simulation of the complete QVGA RC network leads to dynamics around 3μs for the nMOS lattice and 5μs for the pMOS lattice. This is why the ratio W/L of the switches is as high as possible, compromising unfortunately area and accuracy.

[†]The error presents dependency with the block size due to the increasing ratio 'number of switches per cell capacitor' as blocks grow. For small blocks, a significant number of cells with respect to the total amount composing the block will be located at the edges. For example, all cells in a 2×2-px block fall at edges, featuring connectivity to only one neighbor. This means that there is only one switch per capacitor causing charge injection. The error is smaller in this case than when more switches per capacitor are involved in the operation. Indeed, the larger the block size is, the more full-connected cells —that is, cells not located at edges— there will be within the block. This progressively increases the ratio above defined, what in turn increases the error originated by charge injection. Fortunately, this increase rapidly slows down, as shown in Table II. The results presented in this table make total sense in numerical terms since, in the extreme case of an infinite network, there would be two switches per cell capacitor —keep in mind that an infinite 4-connected 2-D lattice can be seen as composed of cells providing connection to the east and south, and therefore receiving connection from the western and northern neighbors. This would mean to double, in principle, the error for small blocks where only one switch per capacitor is accounted. The numerical values extracted from the simulations follow such tendency.

5. HIGH-LEVEL IMPLICATIONS OF EARLY VISION HARDWARE PERFORMANCE

Once the mixed-signal early processing core is characterized, the next step is to evaluate how its moderate computational accuracy affects high-level vision capabilities. This will allow to confirm the validity of our approach. To this end, we make use of the Viola-Jones face detection baseline algorithm provided by the release 2.4.0 of the OpenCV library. As a test bench, we apply the 450 face images composing the Caltech Frontal Face Dataset [34]. These images are scaled down to QVGA resolution to meet our array's specifications. The algorithm is first run without modification in order to obtain the ideal outcome. Each scale is forced to be attained by downsampling the image dimensions of the previous one by a factor 2, just like the proposed processing array operates. Then, the deviation models described in Section 4 are inserted into the Viola-Jones processing flow, as shown in Fig. 13. Note that we comment the function `cvIntegral` available in OpenCV for the computation of integral and square integral images. This function is substituted by two 'for' loops. The first one creates the array images encoded by $V_{S_{ij}}$ and $V_{SQ_{ij}}$ according to Fig 10 and Eqs. (11), (15) and (17). The second one carries out the computation of the integral images, incorporating the error associated to charge redistribution. The resulting integral images' pixels are stored in `double` variables representing analog voltages. In the final step provided by the functions `scale_sum` and `scale_sq`, they are truncated to an integer within the range [0,255] in order to emulate the A-to-D conversion that would deliver the corresponding digital output flow. The complete source code written for the emulation of the array operation together with the test bench, ideal and array outcomes and a description of the different files involved can be found at www.imse-cnm.csic.es/mondego/IJCTA/. The results are summarized in Table III. The ideal outcome (first row) features a hit rate of 81.5%, detecting 367 faces out of 450. No false positives are triggered. Regarding the emulated array outcome (second row), we realized 10 executions of the algorithm in order to take into account the statistical nature of the deviations introduced. The figures presented in Table III correspond to the mean and standard deviation for each parameter. Surprisingly enough [35], the hit rate is slightly higher than for the ideal case, 86.7%, though it is achieved at the cost of some false positives. Finally, in order to account for other possible deviations not considered in our models, we have added white Gaussian

	Faces detected # (out of 450)	**False positives #**	**Hit rate (%)**
Ideal outcome	367	0	81.5
Mixed-signal array	390.2	4.3	86.7
σ	4.1	1.2	1.2
Mixed-signal array (additional noise)	326.9	1.1	72.7
σ	4.9	0.7	1.1

Table III. Comparison between the ideal outcome (first row) and 10 executions of the algorithm including physical deviation models (second row) and an additional source of white Gaussian noise (third row).

noise directly on the pixel value, that is, on V_{ij}. This noise features a mean value of 0V and a standard deviation of 5% of signal range (that is, 45mV). It impacts the operation of pixel copy across the array in an uncorrelated way, affecting therefore squaring and charge redistribution too. The third row in Table III shows the performance of the algorithm when this preemptive deviation is introduced into the processing flow. Despite having considered the worst-case for each primitive and the additional noise, the algorithm still performs relatively well, not far from the ideal case. We can therefore conclude, according to the proposed hardware-software design loop, that there is still margin for a more aggressive design of the mixed-signal processing hardware in terms of area and power consumption at the cost of some extra inaccuracy. Conversely, some parameters of the algorithm could be adjusted for better exploitation of the performance expected from the early vision hardware.

Finally, to conclude this section, we must highlight that the array reaches the performance just described while featuring one of the main assets of the focal-plane sensing-processing approach: power efficiency. Bear in mind that we are targeting typical video frame rates, that is, around 30fps. At this rate, the switching power associated to the peripheral digital circuitry, measured by a magnitude of μW/MHz, will hardly impact energy consumption. Regarding the mixed-signal core, squaring and charge redistribution do not require extra energy once the initial voltages at the corresponding capacitors have been set. Thus, three major sources of power consumption are left: reset of the photodiode and sensing capacitance, precharge of the capacitance holding $V_{SQ_{ij}}$ and pixel copy operation. We will be considering a fixed frame rate of 30fps for the figures provided next. The reset of the photodiode is needed only once per frame. According to the simulations, this operation demands 16.4pW per cell, 1.26μW for the whole array. Regarding precharge for subsequent pixel squaring, a single operation requires only 0.18pJ. However, it is performed at each cell of the array for each pixel of the square integral image at each scale. Adding up all these operations for five scales, the resulting power consumption is 42.4mW. Likewise, a single copy of pixel demands only 0.47pJ whereas all the operations required across five scales amount to 111.36mW. A total power consumption always less than 200mW is therefore expected for the mixed-signal processing array. As a reference, we can scale this figure in order to compare it to the power consumption reported in [27]: 240mW for a smart camera handling 30×30-px images at 80fps. Under these conditions of image size and frame rate, the power consumption of our array boils down to less than 100μW. Of course this comparison is not fair enough since the camera described in [27] constitutes a general-purpose digital system carrying out the complete Viola-Jones processing flow. However, it still permits to give an idea of the energy efficiency reached by the approach presented. Starting at 100μW for imaging and low-level processing, it seems rather feasible to address the design of a vision system featuring a power consumption significantly less than 240mW.

6. CONCLUSIONS

The design of smart image sensors for artificial vision application frameworks must be driven by the vision algorithm to be implemented. A comprehensive analysis of the characteristics of such algorithm will permit to convey to the sensing-processing stage just those operations where the smart sensor can really become efficient. In this paper, we demonstrate the physical and functional feasibility of a QVGA mixed-signal processing array tailored for the Viola-Jones processing framework. We study the ultimate consequences of physical non-idealities on

```
cvResize( img, &img1, CV_INTER_LINEAR );
// cvIntegral( &img1, &sum1, &sqsum1, _tilted );

// ////////////////////////////////////////////////////////////////////////////
// Code introduced to emulate the performance expected from the mixed-signal array

srand(time(NULL)); // initialize random number generator

// We first create the images from which the integral images are going to be computed
for (rows = 0; rows < img1.rows; rows++)
{
    for (cols = 0;cols < img1.cols; cols++)
    {
        orig_pixel = *CV_MAT_ELEM_PTR_FAST(img1,rows,cols,CV_ELEM_SIZE(img1.type)); // digitized ideal pixel value
        // conversion to array signal range, mismatch between sensing capacitances and charge injection error
        array_pixel = cap_mismatch_ch_injection(orig_pixel);
        // noise added to the "captured" image in order to account for other possible sources or error
        array_pixel += (MAX_PX - MIN_PX)*randn(0,STD_ADD)/100;
        array_copy = buffer(array_pixel); // error in pixel copy
        array_sq = squarer(array_copy); // error in squaring
        *(double *)CV_MAT_ELEM_PTR_FAST(array_image1,rows,cols,sizeof(double)) = array_copy;
        *(double *)CV_MAT_ELEM_PTR_FAST(array_sqimage1,rows,cols,sizeof(double)) = array_sq;
    }
}

// Next the integral images are computed
for (rows = 0; rows < img1.rows; rows++)
{
    for (cols = 0; cols < img1.cols; cols++)
    {
        array_sum = 0;
        array_sqsum = 0;
        for (i = rows; i >= 0; i--)
        {
            for (j = cols; j >= 0; j--)
            {
                array_copy = *(double *)CV_MAT_ELEM_PTR_FAST(array_image1,i,j,CV_ELEM_SIZE(array_image1.type));
                array_sq = *(double *)CV_MAT_ELEM_PTR_FAST(array_sqimage1,i,j,CV_ELEM_SIZE(array_sqimage1.type));
                array_sum += array_copy; // integral sum
                array_sqsum += array_sq; // square integral sum
            }
        }
        // We accommodate the signal range back to digital values. The offset due to charge redistribution is also added here
        *(sumtype*)CV_MAT_ELEM_PTR_FAST(sum1,rows+1,cols+1,sizeof(sumtype)) = scale_sum(array_sum,rows+1,cols+1,sz.height,sz.width);
        *(sqsumtype*)CV_MAT_ELEM_PTR_FAST(sqsum1,rows+1,cols+1,sizeof(sqsumtype)) = scale_sq(array_sqsum,rows+1,cols+1,sz.height,sz.width);
        if (rows == 0 || cols == 0)
        {
            *(sumtype*)CV_MAT_ELEM_PTR_FAST(sum1,rows,cols,sizeof(sumtype)) = 0;
            *(sqsumtype*)CV_MAT_ELEM_PTR_FAST(sqsum1,rows,cols,sizeof(sqsumtype)) = 0;
        }
    }
}

*(sumtype*)CV_MAT_ELEM_PTR_FAST(sum1,img1.rows,0,sizeof(sumtype)) = 0;
*(sumtype*)CV_MAT_ELEM_PTR_FAST(sum1,0,img1.cols,sizeof(sumtype)) = 0;
*(sqsumtype*)CV_MAT_ELEM_PTR_FAST(sqsum1,img1.rows,0,sizeof(sqsumtype)) = 0;
*(sqsumtype*)CV_MAT_ELEM_PTR_FAST(sqsum1,0,img1.cols,sizeof(sqsumtype)) = 0;

// ////////////////////////////////////////////////////////////////

int ystep = factor > 2 ? 1 : 2;
```

Figure 13. Source code emulating the array operation inserted into the Viola-Jones processing flow provided by the OpenCV libray.

the high-level interpretation of the scene targeted by this framework. The presented results definitely prove the validity of our approach and highlight its major asset: energy efficiency.

REFERENCES

1. Fossum ER. CMOS image sensors: electronic camera-on-a-chip. *IEEE Trans. Electron Devices* 1997; **44**(10):1689–1698.
2. Ohta J. *Smart CMOS Image Sensors and Applications*. CRC Press, 2008.
3. Zarándy A Ed. *Focal-plane Sensor-Processor Chips*. Springer, 2011.

4. Tomasi C. Early vision. In *Encyclopedia of Cognitive Science*. John Wiley & Sons Ltd, 2006.
5. Unger SH. A computer oriented toward spatial problems. *Proceedings of the IRE* 1958; **46**(10):1744–1750.
6. Nilchi A, Aziz J, Genov R. Focal-plane algorithmically-multiplying CMOS computational image sensor. *IEEE J. Solid-State Circuits* 2009; **44**(6):1829–1839.
7. Jendernalik W, Blakiewicz G, Jakusz J, Szczepanski S, Piotrowski R. An analog sub-miliwatt CMOS image sensor with pixel-level convolution processing. *IEEE Trans. Circuits Syst. I* 2013; **60**(2):279–289.
8. Fernandez-Berni J, Carmona-Galan R. All-MOS implementation of RC networks for time-controlled Gaussian spatial filtering. *Int. J. of Circuit Theory and Applications* 2012; **40**(8):859–876.
9. Gottardi M, Massari N, Jawed SA. A 100μW 128×64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications. *IEEE J. Solid-State Circuits* 2009; **44**(5):1582–1592.
10. Leñero-Bardallo JA, Serrano-Gotarredona T, Linares-Barranco B. A 3.6μs latency asynchronous frame-free event-driven dynamic-vision-sensor. *IEEE J. Solid-State Circuits* 2011; **46**(6):1443–1455.
11. Lichtsteiner P, Posch C, Delbruck T. A 128×128 120dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid-State Circuits* 2008; **43**(2):566–576.
12. Cottini N, Gottardi M, Massari N, Passerone R, Smilansky Z. A 33μW 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation. *IEEE J. Solid-State Circuits* 2013; **48**(3):850–863.
13. Oliveira F, Haas H, Gomes J, Petraglia A. CMOS imager with focal-plane analog image compression combining DPCM and VQ. *IEEE Trans. Circuits Syst. I* 2013; **60**(5):1331–1344.
14. Messaoud FG, Peizerat A, Dupret A, Blanchard Y. On-chip compression for HDR image sensors. *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP)* 2010.
15. Foldesy P, Zarándy A, Rekeczky C. Configurable 3D-integrated focal-plane cellular sensor-processor array architecture. *Int. J. of Circuit Theory and Applications* 2008; **36**(5-6):573–588.
16. Lopich A, Dudek P. Asynchronous cellular logic network as a co-processor for a general-purpose massively parallel array *Int. J. of Circuit Theory and Applications* 2011; **39**(9):963–972.
17. Rodríguez-Vázquez A, Domínguez-Castro R, Jímenez-Garrido F, Morillas S, Listán J, Alba L, Utrera C, Espejo S, Romary R. The Eye-RIS CMOS vision system. In Chapter 2, *Sensors, Actuators and Power Drivers; Integrated Power Amplifiers from Wireline to RF; Very High Frequency Front Ends* Springer, 2008;
18. Wyatt JL, Keast C, Seidel M, Standley D, Horn B, Knight T, Sodini C, Hae-seung Lee, Poggio T. Analog VLSI systems for image acquisition and fast early vision. *Int. J. of Computer Vision* 1992; **8**(3):217–230.
19. Viola P, Jones M. Robust real-time object detection. *Proceedings of the 2nd Int. Workshop on Statistical and Computational Theories of Vision* 2001.
20. Viola P, Jones M. Robust real-time face detection. *Int. J. of Computer Vision* 2004; **57**(2):137–154.
21. Bradski G. The OpenCV library. *Dr. Dobb's Journal of Software Tools* 2000; http://opencv.org/
22. Ngo H, Rakvic R, Broussard R, Ives R. An FPGA-based design of a modular approach for integral images in a real-time face detection system. *Proceedings of the SPIE Mobile Multimedia/Image Processing, Security, and Applications* 2009.
23. Das S, Jariwala A, Engineer P. Modified architecture for real-time face detection using FPGA. *Proceedings of the Nirma University Int. Conf. on Engineering* 2012.
24. Acasandrei L, Barriga A. FPGA implementation of an embedded face detection system based on LEON3. *Proceedings of the World Congress in Computer Science, Computer Engineering and Applied Computing* 2012.
25. Hefenbrock D, Oberg J, Thanh N, Kastnert R, Baden S. Accelerating Viola-Jones face detection to FPGA-Level using GPUs. *Proceedings of the 18th IEEE Int. Symp. on Field-Programmable Custom Computing Machines* 2010.
26. Haipeng Jia, Yunquan Zhang, Weiyan Wang, Jianliang Xu. Accelerating Viola-Jones face detection algorithm on GPUs. *Proceedings of the IEEE Int. Conf. on Embedded Software and Systems* 2012.
27. Camilli M, Kleihorst R. Demo: mouse sensor networks, the smart camera. *Proceedings of the 5th ACM/IEEE Int. Conf. on Distributed Smart Cameras* 2011.
28. Acasandrei L, Barriga A. Accelerating Viola-Jones face detection for embedded and SoC environments. *Proceedings of the 5th ACM/IEEE Int. Conf. on Distributed Smart Cameras* 2011.
29. Mallat SG. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 1989; **11**(7):674–693.
30. Fernández-Berni J, Carmona-Galán R, Rodríguez-Vázquez Á. Reconfigurable focal-plane hardware for block-wise intra-frame HDR imaging. *Proceedings of the Int. Image Sensor Workshop* 2013.
31. Fernández-Berni J, Carmona-Galán R, Carranza-González L. FLIP-Q: A QCIF resolution focal-plane array for low-power image processing. *IEEE J. Solid-State Circuits* 2011; **46**(3):669–680.
32. Fernández-Berni J, Carmona-Galán R, Rodríguez-Vázquez Á. Image filtering by reduced kernels exploiting

kernel structure and focal-plane averaging. *Proceedings of the IEEE European Conf. on Circuit Theory and Design (ECCTD)* 2011.

33. Bo Zhao, Xiangyu Zhang, Shoushun Chen A 64×64 CMOS image sensor with on-chip moving object detection and localization. *IEEE Trans. on Circuits and Systems for Video Technology* 2012; **22**(4):581–588.
34. "Caltech Frontal Face Dataset", http://www.vision.caltech.edu/html-files/archive.html
35. Kundu A, Sarkar S. A possible model of noise enhanced visual perception in human vision. *Proceedings of the Int. Conf. on Scientific Paradigm Shift in Information Technology and Management* 2011.