A Micropower Centroiding Vision Processor

Timothy G. Constandinou, *Member, IEEE*, and Christofer Toumazou, *Fellow, IEEE*

***Abstract*—A biologically-inspired hybrid vision chip is presented for real-time object-based processing for tasks such as centroiding, sizing and counting of enclosed objects. This system presents the first silicon retina capable of centroiding and sizing multiple objects in true parallel fashion. Based on a novel distributed algorithm, this approach uses the input image to enclose a feedback loop to realize a data-driven pulsating action. The sensor provides a resolution of 48 × 48 pixels with a 85 μm × 85 μm pixel footprint and has been measured to consume 243 μW at 1.8-V supply, achieving an equivalent computational efficiency of 724.64 MIPS/mW with a 500-μs process time.**

***Index Terms*—Centroiding, CMOS image sensor, distributed algorithm, focal plane processing, micropower, object detection, parallel processing, silicon retina, target tracking, vision chip.**

I. Introduction

MODERN advanced image processing systems use an external camera to stream image data to the processor executing the software algorithm. Such a modular scheme demands a huge communication bandwidth for the video transmission and, therefore, heavy power requirements. Several early filtering applications can benefit from combining the phototransduction and processing at the pixel level. A new breed of vision chips have recently emerged that strive to achieve precisely this. A generic reconfigurable architecture to provide such pixel-level processing is the cellular neural network (CNN) processor [1]. Other systems have been inspired by the unparalleled computational efficiency of living organizms in solving complex image processing tasks. These biologically-inspired (or retinomorphic [2]) systems have been realized to perform tasks such as image enhancement and feature extraction.

Systems that require image recognition functionality, such as target tracking and centroid detection are relatively demanding to implement and require certain perceptive or high-level behavior. Traditional image processing techniques effectively perform low-level tasks such as conditioning and filtering but typically output a matrix of pixels, constituting an image. For perceptive vision applications it is paramount to cluster together pixels in a region of interest and provide a single entity. This task is often referred to as object segmentation. Having performed this, it is useful to define the object using a coordinate and magnitude to represent its centroid and size, respectively. Having such information available can provide enhanced and added functionality to several applications. For example, providing navigational cues for position and bearing in space exploration (i.e., in autonomous navigation), solar centroiding for tilt/positioning of solar panels, and image stabilization during motion (by target tracking) are some applications that could benefit from advances in such processing techniques. For mobile platforms, including autonomous systems and handheld devices, minimizing power consumption is of the upmost importance.

Distributed vision processing provides a method to implement such computationally demanding algorithms with good power efficiency. As already mentioned, by distributing the processing throughout the focal plane, traditional data communication bottlenecks can be overcome. Therefore, such techniques can be used to deliver high framerate processing at good computational efficiency, for a certain area penalty.

Past work (see Table I for comparison) in this field has focused mainly on implementing the center-of-mass (COM) computation in distributed hardware. Typically, this technique provides high (multiple sub-pixel) accuracy, high speed, good robustness and compact hardware implementation. However, systems based on this technique are inherently limited to single object centroiding and without any capability for size computation. Komuro *et al.* [3] have reported the only such system capable of tracking multiple centroids. Here, the method adopted uses a combination of binarization and COM computation in the focal-plane, with the multiple target capability being implemented off-chip using a *self-windowing* algorithm.

In this work, we present a novel vision chip for multiple object center and size processing of simple uniform objects. This is the first system reported to achieve true parallel position detection of multiple objects in a focal-plane processor. This system performs object segmentation by local thresholding and then size and centroid computation by object boundary reduction timing and convergence detection, respectively. The processed centroid and size is determined within a ± 1 pixel accuracy. The circuit implementation uses a hybrid analog and asynchronous digital pixel core, realising a computationally efficient vision processor delivering 724.64 MIPS/mW. Furthermore, the parallel architecture results in a high computational capacity, achieving a process time of under 500 μs.

This paper is organized as follows. In Section II, the developed algorithm is described and the computation, robustness, accuracy and feasibility for hardware implementation are discussed. Section III outlines the general system architecture and Section IV describes the circuit-level design and implementation. In Section V, the 0.18-μm CMOS prototypes are described and the measured results are presented. Finally, in Section VI, we discuss and summarize the performance of the centroiding vision processor.

Manuscript received August 20, 2005; revised January 2, 2006. This work was supported by a Basic Technology grant (UKRC GR/R87642/02) and an AMx Technology grant (EPSRC GR/R96583/01), and by Toumaz Technology Limited.

The authors are with the Institute of Biomedical Engineering, Imperial College of Science, Technology and Medicine, South Kensington Campus, London SW7 2AZ, U.K. (e-mail: t.constandinou@imperial.ac.uk).

Digital Object Identifier 10.1109/JSSC.2006.874330

TABLE I
COMPARATIVE REVIEW OF CENTROID DETECTING VISION CHIPS

Year	Ref.	Tech. (μm)	Die Size (mm^2)	Array Size	Pixel Size (μm^2)	Pixel Format	Fill Factor(%)	Multiple Objects	Accuracy (Pixels)	Speed (FPS)	Power (mW)
2004	[4]	0.5	-	43×43	207	APS	49	no	0.3	-	-
2004	[5]	0.7	18.00	5×5	100	Analogue	-	no	-	4800	-
2004	[6]	0.8	6.25	20×20	204	Analogue	12	no	0.013	3000	15
2003	[7]	0.18	6.00	80×80	6900	Binary	51	no	0.1	1000	30
2003	[3]	0.5	49.00	64×64	156	Binary	18.8	18 max.	1	1000	112
2003	[8]	0.6	20.25	11×11	25	Binary	6.7	yes	0.1	10000	-
2002	[9]	0.5	2.25	24×24	1111	Analogue	18.7	no	-	1000	-
2002	[10]	0.6	2.88	60×36	514	Analogue, APS	12, 16	no	1	3600	2.6
1999	[11]	0.5	20.25	43×43	201	Binary	23	no	-	-	-
1999	[12]	1.2	4.00	40×1	12744	Analogue	5	no	-	8000	-
1999	[13]	1.2	22.09	25×24	25472	Analogue	3	no	-	-	19.9
1999	[14]	2.0	4.93	23×1	89050	Analogue	-	no	-	-	5.4
1998	[15]	2.0	6.00	24×24	3844	Analogue	30	no	3	7000	0.25
1998	[16]	3.0	-	25×1		Neuron	-	no	1.5	-	3

II. METHOD

The distributed algorithm presented is intended for object-based processing including centroiding and sizing of simple objects [17]. Circular *blob*-like objects with uniform texture and an intensity differing from the background level can be segmented and their size and position (centroid) determined by means of a distributed binary algorithm.

This uses an edge-detection technique to form the contours and trigger the data-driven processing. On detection of an object boundary, the initial state for the signal flow is set. By propagating an inward fill, the contour can be reduced until it converges to the center. The central point is detected by utilising spatiotemporal integration. On convergence, the object is reset and output transmitted, thus realising an inward pulsating action. Furthermore, the frequency of pulsation can be directly used to determine the object size.

The functionality of this algorithm is divided into two groups: image processing operations (for feature extraction) and binary processing operations (for detail extraction).

A. Image Processing

The overall image processing functionality for feature extraction is illustrated in Fig. 1.

1) Thresholding: Objects are defined as regions in the image with narrow-field (2 × 2 pixel) local-average intensity either below (or above) the average background intensity. In a distributed fashion, this requires a single discrete signal (THRESHOLD) be generated per pixel. This signal is used to facilitate the object segmentation by defining the valid area the signal can propagate within. As the centroid processing occurs at the pixel corners, a local average, centerd on that node is required to represent the intensity at that point. This is defined in (1):

$$Av_{\text{narrow}} = \frac{1}{4}\sum_{i=x}^{x+1}\sum_{j=y}^{y+1} I_{i,j} \tag{1}$$

where $I_{i,j}$ is the intensity of pixel i, j, and x and y are the array dimensions.

Similarly, to compute the global average, this can be extended to cover the entire array. However, for images with varying background intensity, it is favorable to use a *wide-field* local average, covering a large enough area to extend beyond a single object. An easily hardware-implementable scheme is to average across the entire column and/or row, providing a "global" average unique to every pixel, as expressed in (2):

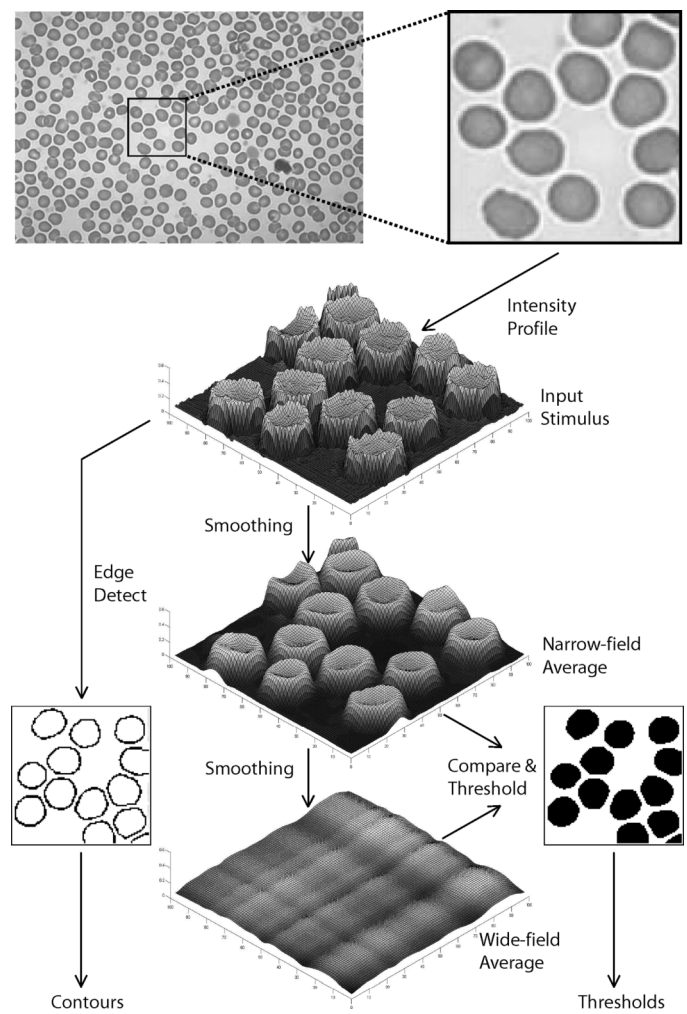


Fig. 1. Front-end continuous-time image-processing functionality.

$$Av_{\text{wide}} = \frac{1}{2x}\sum_{i=1}^{x} I_{\text{narrow}(i,j)} + \frac{1}{2y}\sum_{j=1}^{x} I_{\text{narrow}(i,j)}. \tag{2}$$

The THRESHOLD signal is then obtained by comparing the wide and narrow field averages. To provide some tolerance to image noise and component mismatch, an artificial offset is applied to the comparator.

2) Edge Detection: The edges are detected by comparing all adjacent pixel intensities in a vertical or horizontal direction and flagging a CONTOUR if a cell has two edges leading to it, i.e., a continuous edge. This can be defined as a Boolean expression, given in (3):

$$Co = \overline{(Th)} \cdot \overline{(\overline{A} \cdot \overline{B} \cdot \overline{C}) + (\overline{A} \cdot \overline{B} \cdot \overline{D}) + (\overline{A} \cdot \overline{C} \cdot \overline{D}) + (\overline{B} \cdot \overline{C} \cdot \overline{D})} \quad (3)$$

where A, B, C, and D are the four edge inputs and Th is the threshold status.

B. Binary Processing

Having extracted two discrete signals per pixel: CONTOUR and THRESHOLD, these are used to feed, control, and regulate the asynchronous *neuronal* network.

To facilitate the pulsating action, each cell requires three bits of static memory: STATE, RESET, and CENTRE. These are updated asynchronously depending on a cell's CONTOUR and THRESHOLD inputs, its current STATE, RESET, and CENTRE values, and those received from surrounding cells. Therefore, the internal functionality of a binary processing element can be described using Boolean expressions driving a state machine.

The pulsating action is initiated by setting a cell's state on detection of a CONTOUR. The reduction is facilitated by checking whether any neighboring cells have their STATE asserted in addition to the object criterion (THRESHOLD) being satisfied. The rate of this contour reduction is defined by an artificial propagation delay introduced in this event path. The set STATE (S_{Set}) condition is defined in (4):

$$S_{\mathrm{Set}}(t+\tau_s) = Co(t) + Th(t) \cdot [S_1(t) + S_2(t) + S_3(t) + S_4(t)] \quad (4)$$

where τ_s is the delay time-constant that defines the propagation rate, $Co(t)$ is the current CONTOUR status, $Th(t)$ is the current THRESHOLD status, and $S_1(t), S_2(t), S_3(t), S_4(t)$ are the STATE variables of the directly adjacent cells.

The reset STATE (S_{Reset}) and set RESET (R_{Set}) events occur under identical conditions, occurring either on CENTRE being reset, or on a RESET back-propagation (from adjacent cells) during a local reset. This condition is defined in (5):

$$S_{\mathrm{Reset}}(t+\delta t) = C(t-\delta t) \cdot \overline{C}(t) + S(t) \cdot [R_1(t) + R_2(t) + R_3(t) + R_4(t)] \quad (5)$$

where δt represents the propagation delay of the combinational logic, $S(t)$ is the cell's current STATE value, $C(t)$ is the current CENTRE status, and $R_1(t), R_2(t), R_3(t), R_4(t)$ are the RESET variables of the directly adjacent cells.

In parallel with the contour reduction, each pixel checks whether it satisfies the centroid detection criteria. This is defined when surrounding cells (but not directly adjacent) have been asserted but the centroid pixel has not. This condition flags a CENTRE signal [defined in (7)], transmitting the pixel's coordinate and issuing a local RESET signal. Furthermore, lateral inhibition [eq. (6)] prevents a cell flagging a CENTRE if any adjacent cell has its CENTRE asserted.

$$C_{\mathrm{Inhibit}}(t+\delta t) = C1(t) + C2(t) + C3(t) + C4(t) + C5(t) + C6(t) + C7(t) + C8(t) \quad (6)$$

where $C1(t)$ to $C8(t)$ are the CENTRE status of the four directly- and four diagonally-adjacent cells.

$$C_{\mathrm{Set}}(t+\delta t) = \overline{C_{\mathrm{Inhibit}}} \cdot \overline{S(t)} \cdot S_{12}(t) \cdot S_{22}(t) \cdot S_{32}(t) \cdot S_{42}(t) \quad (7)$$

where $S_{12}(t)$, $S_{22}(t)$, $S_{32}(t)$, and $S_{42}(t)$ are the STATE values of cells two to the left, two above, two to the right, and two below, respectively.

This CENTRE becomes reset when the contour reduction finally reaches the center cell, defined in (8):

$$C_{\mathrm{Reset}}(t+\delta t) = \overline{S(t)} \cdot S_1(t) \cdot S_2(t) \cdot S_3(t) \cdot S_4(t) \quad (8)$$

where $S_1(t)$, $S_2(t)$, $S_3(t)$, and $S_4(t)$ are the STATE values of the directly adjacent cells.

This RESET signal is then back-propagated outwards in a recursive manner, similar to the contour reduction with the absence of the artificial delay. This delay would be undesirable in the reverse path, to avoid flagging multiple centroids. The RESET memory is configured to self-reset, i.e., operate as a monostable.

An unusual but important feature of this method is the absence of any pre-defined synchronization signal, for example, a clock. The only synchronization is obtained through the data-driven object reset scheme, but on a local, rather than a global basis. This in combination with the artificial delay time-constant defines the processing time.

C. Simulation

Being an algorithm of both distributed and asynchronous nature, it is both complex to model and computationally demanding to simulate in its precise form. However, a "frame-based" representation can be derived by making some basic assumptions. Primarily, all delay elements are assumed to be perfectly matched, therefore allowing all pixels to be processed sequentially for each delay "period". The full source code for this implementation is available from [18].

1) Effective Computation: To estimate the effective computation of this distributed algorithm, the frame-based equivalent algorithm is evaluated. Although many of the functions are static, there also exist dynamic functions, for example the reset cycle being recursive. The static computation is dependant only on the array dimensions, whereas dynamic computation is largely dependant on input data.

For a pixel array of dimensions (x, y), with n objects ($a \in [0, n]$) of radius r_n pixels each, the number of processor instructions per frame is determined [18]. Assuming a frame capture and process time of t μs, the complete computational load is given by

$$C = \frac{1 + 8y + 37xy + 6\sum_{a=1}^{n}\left(\pi r_a^2\right)}{t \times 10^{-6}}. \quad (9)$$

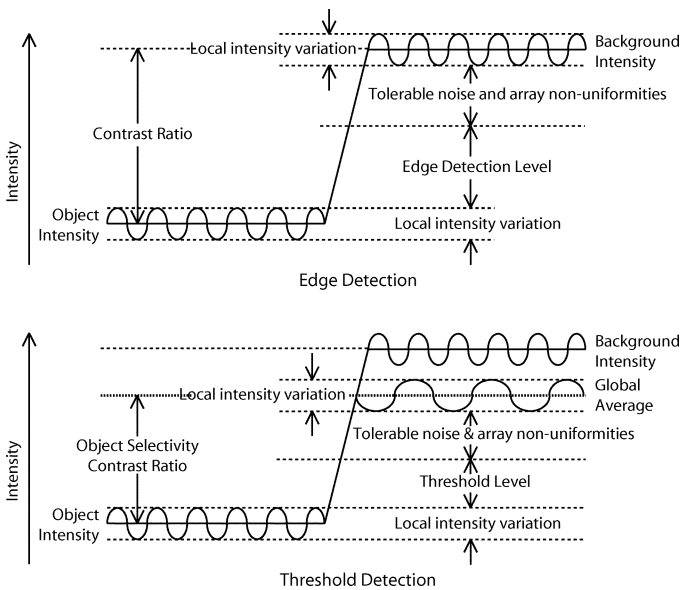


Fig. 2. Acceptable noise margins for error-free binary edge detection (top) and object thresholding (bottom).

D. Robustness

The *robustness* of the algorithm provides an indication to its immunity to fuzzy, ill-conditioned, or noisy data. The *yield* expresses the system reliability against fabrication defects and process variations (in custom hardware). In order to analyze and evaluate the robustness of the algorithm, the input image data can be pre-filtered to include array nonuniformities such as fixed-pattern noise, pixel sensitivity, and feature detection fluctuations.

1) Noise Margin: Considering the intensity profiles and error tolerances of the input images, the *noise margin* or *signal-to-noise ratio* can be determined for error-free feature extraction. Subsequently for a given image type, the optimal threshold and edge levels can be deduced for maximum binary robustness and furthermore the suitability of the algorithm to different image types can be analyzed.

For edge detection to be reliable in an input image consisting of (relatively) dark objects on a light background, the conditions specified in (10) (for edge detection) and (11) (for no edge detection) must be satisfied [see Fig. 2 (top)].

$$(I_{\rm bg} - I_{\rm obj}) - E_{\rm offset} > I_{\rm margin} + I_{\rm noise} \tag{10}$$

$$E_{\rm offset} > I_{\rm margin} + I_{\rm noise} \tag{11}$$

where $(I_{\rm bg} - I_{\rm obj})$ is the contrast difference of the objects to background level, $E_{\rm offset}$ is the minimum edge detection level, $I_{\rm margin}$ is the maximum object (and/or background) intensity variation, and $I_{\rm noise}$ represents the maximum tolerable level of intensity variations (nonuniformities) in the array.

Assuming $I_{\rm margin}(background) = I_{\rm margin}(object)$ and $I_{\rm noise}$ is increased to the maximum allowable level such that (10) and (11) become equalities, the optimal setting for $E_{\rm offset}$ can be determined for a given image type as described in (12):

$$E_{\rm offset} = \frac{1}{2}(I_{\rm bg} - I_{\rm obj}). \tag{12}$$

However, the actual robustness to noise [eq. (11)] causing erroneous edge detection is in fact increased by the CONTOUR logic. This operates to effectively screen out any erroneous edges within object boundaries by only detecting edges outside object (THRESHOLD) regions. Furthermore by performing the thresholding operation on the smoothed intensities, the object segmentation is also robust against image noise. For the reliability of the threshold detection to be maximized in an input image consisting of dark objects on a light background, the conditions specified in (13) (for thresholding) and (14) (for no thresholding) need to be satisfied [see Fig. 2 (bottom)].

$$(I_{\rm av} - I_{\rm obj}) - T_{\rm offset} > I_{\rm margin} + I_{\rm noise} \tag{13}$$

$$(I_{\rm bg} - I_{\rm av}) + T_{\rm offset} > I_{\rm margin} + I_{\rm noise} \tag{14}$$

where $(I_{\rm av} - I_{\rm obj})$ is the margin from average intensity to object intensity level, $(I_{\rm bg} - I_{\rm av})$ is the margin from background intensity to average intensity level, $T_{\rm offset}$ defines the threshold detection offset from average intensity, $I_{\rm margin}$ is the maximum object (and/or background) intensity variation, and $I_{\rm noise}$ represents the maximum tolerable level of intensity variations (nonuniformities) in the array.

Assuming $I_{\rm margin}(background) = I_{\rm margin}(object)$ and $I_{\rm noise}$ is increased to the maximum allowable level such that (13) and (14) become equalities, the optimum setting for $T_{\rm offset}$ can be specified for a given image type as expressed in (15):

$$T_{\rm offset} = \frac{1}{2}(2I_{\rm av} - I_{\rm obj} - I_{\rm bg}) \tag{15}$$

2) Distributed Processing Robustness: A major objective in implementing distributed algorithms employing massive parallelism and thus redundancy is to boost the robustness, defect immunity, and tolerance to ill-conditioned data. Subsequently, this algorithm is shown to substantially increase its robustness beyond the expected analytical feature extraction limit. This can be attributed to the parallel or distributed processing forming multiple data flow paths coupled with data redundancy and compression. In context, the data flow is initiated at the object contours and an inward fill is facilitated. This has the effect of compressing the amount of data being processed; for the "ring" of cells being processed at any time reduces with this inward propagation. Therefore, this shrinking ring realizes a many-to-one mapping and thus introduces massive data redundancy.

3) Statistical Simulations: One approach to quantify this robustness is through statistical algorithmic simulations. This involves experimentally testing the algorithm for various input images against the expected sources of error. To facilitate this, the input image is pre-filtered to include various levels of pixel-array nonuniformities, testing the algorithm for each data set. The procedure taken is as follows.

- Input image: Ten different input images are used of flat object and background texture.
- Contrast ratio: Each image is adjusted to three contrast ratios (object to background) of 10:1, 4:1 and 3:2.
- Noise types: The sample images are subjected to Gaussian (fixed pattern), speckle (gain error), and salt/pepper noise (low/high response).

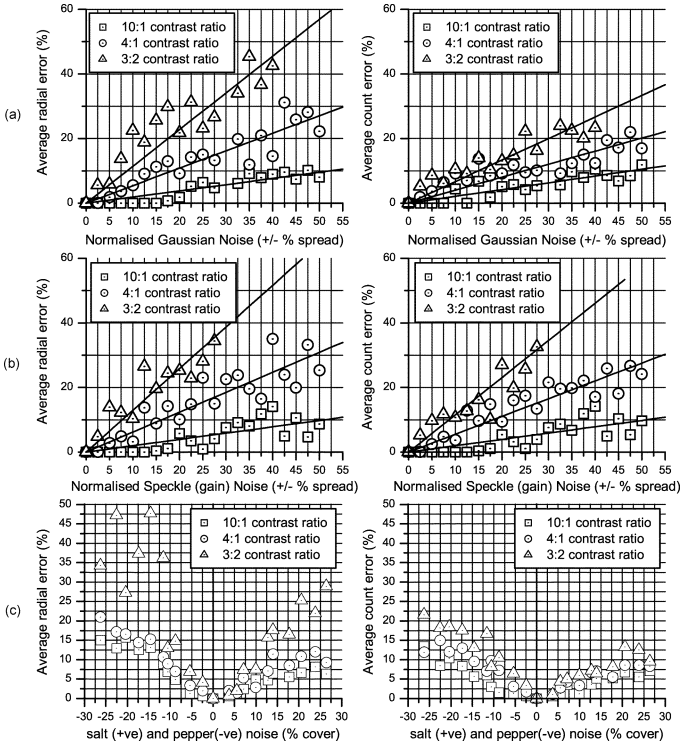


Fig. 3. Statistical simulations to demonstrate robustness to array nonuniformities. Algorithmic response to: (a) additive spacial Gaussian noise (representing fixed pattern noise), (b) random speckle noise (representing array gain/sensitivity nonuniformity), and (c) salt-and-pepper noise representing pixels with permanent low/high response.

- Noise power: Each noise type is tested at 20 levels of noise power to cover $\pm 50\%$.
- Multiple simulation: Each simulation is repeated using ten different noise sets.
- Algorithm settings: For each image/contrast ratio a single set of edge/threshold levels are selected based on (12) and (15).
- Post simulation analysis: Results are averaged, normalized to image content, and aggregated.

The analyzed results are illustrated in Fig. 3 for Gaussian, speckle, and salt/pepper noise. The inherent robustness of this algorithm is demonstrated, showing no point at which the algorithm ceases to operate. However, the centroid or size accuracy becomes degraded with increased image noise. For typical fixed pattern noise levels [19] and medium contrast selectivity, a 2%–5% inaccuracy can be observed and for images with higher contrast ratios, higher accuracies can be expected. The algorithm also proves to be robust to defective pixel outputs, as indicated in the salt-and-pepper noise results.

E. Accuracy

Due to the binary nature of this centroiding/sizing algorithm, the accuracy for regular (circular) objects is limited to single pixel resolution. This, however, deteriorates for irregular objects. This algorithm is intended to provide a good estimate to centroid position and size, not a mathematically accurate center-of-mass computation.

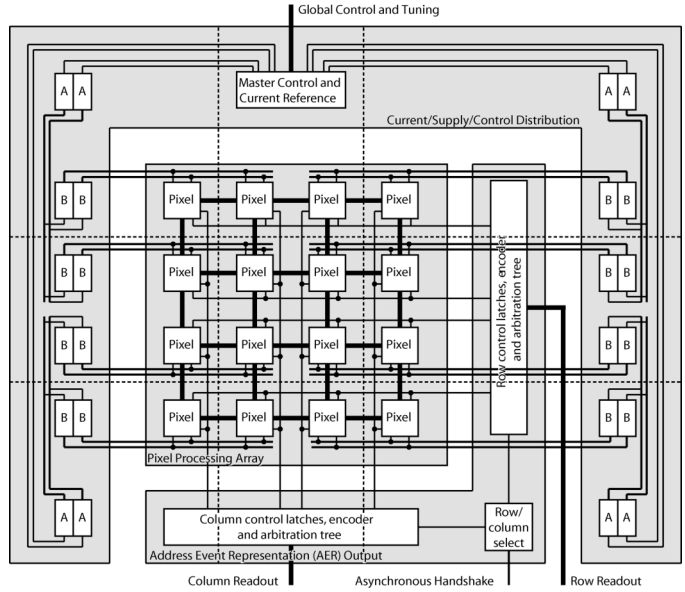


Fig. 4. The proposed system architecture. Illustrated are the three main components: pixel processing array, address event representation readout, and current/supply/control distribution tree. The dotted lines represent the "stretch" marks, i.e., how the system can be scaled to a larger size array.

III. System Organization

This section outlines the adopted system organization for hardware implementation of the presented algorithm. The implemented system architecture [17] consists of three main sub-systems: the processing array (to execute the distributed algorithm), the bias distribution (to provide control and tuning to the array), and the address-event readout (to transmit the results off-chip). This organization is illustrated in Fig. 4.

A. Array Organization

Due to the large number of in-pixel *processing elements*, a current distribution scheme is required for reliable bias current duplication and a hierarchical fanout is required for distribution of digital control signals. For the given size of the pixel array (48×48), the following distribution tree is proposed for both the current and control distribution:

1) Corner (1 to 4): Four identical master bias currents are generated and used to supply the four corner headers of the pixel array (shown in Fig. 4: sub-block A).
2) Row (1 to 24): Each master reference set is used to generate (y/2) copies feeding every row header in its corner (shown in Fig. 4: sub-block B).
3) Column (1 to 24): Each row header is used to generate $(x/2)$ copies feeding every pixel in its row.
4) Pixel (1 to 4): Within each pixel the bias currents are locally combined and duplicated further.

System Scalability: The described architecture allows for a certain degree of scalability (up to an order of magnitude), however, to scale to much larger arrays, the distribution hierarchy (duplication levels) would have to be extended. The pixel circuitry and address-event hardware are, however, both fully scalable.

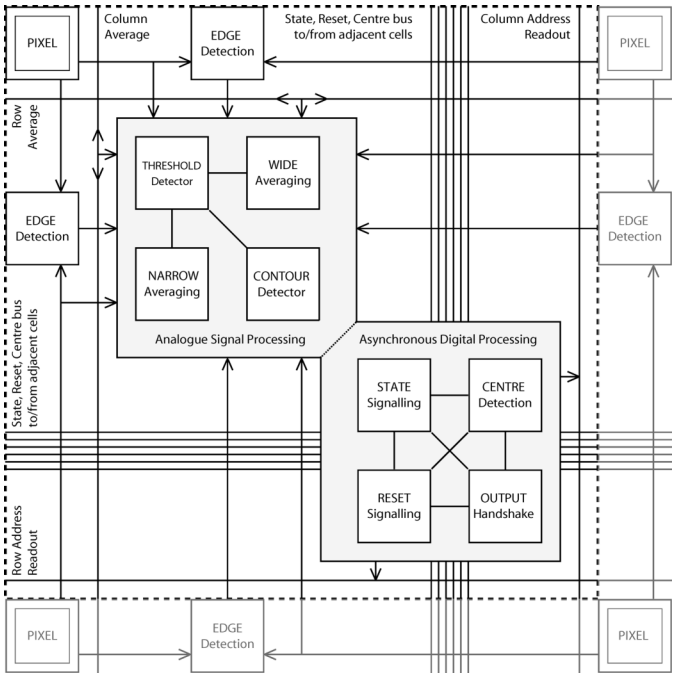


Fig. 5. Proposed cellular architecture for object-based processing illustrating organization and connectivity of functional blocks within a quad-pixel arrangement.

Current-mode Versus Voltage-Mode Current Distribution: In a current-mode scheme, at each current-distribution chain, the currents are copied locally using devices in close proximity (therefore, well-matched) and then distributed via separate metal lines. This increases both device count and metal area utilization in comparison to a voltage-mode current distribution scheme. Such a scheme uses voltage distribution to set device input voltages (over a large area) used to create the bias currents. However, here the error contribution is two-fold; systematic in addition to increased mismatch. Using set bias voltage distribution, metal line resistance over a relatively large distance results in a linear voltage gradient. When used to generate bias currents, this translates in a nonlinear current gradient. Furthermore the mismatch increases due to large proximity device separation where process variation gradients come into effect. For the above mentioned reasons, current-mode current distribution is chosen for this particular application, aiming to achieve relatively good current matching.

B. Pixel Organization

The basic pixel organization is illustrated in Fig. 5. Based on the algorithm described previously in Section II, this architecture realizes a direct implementation.

A reverse-biased parasitic p-n junction is used to generate the photocurrent. This feeds the in-pixel analog signal processing (ASP) core, which smooths, averages, compares, and thresholds, as described earlier. The ASP generates two signals, CONTOUR and THRESHOLD, which in turn are used to feed the asynchronous binary processing (ABP) core. This facilitates the contour reduction through asynchronous signal propagation, flagging the centers on detection. The output neuron then negotiates the address event bus for a timing slot for off-chip communication.

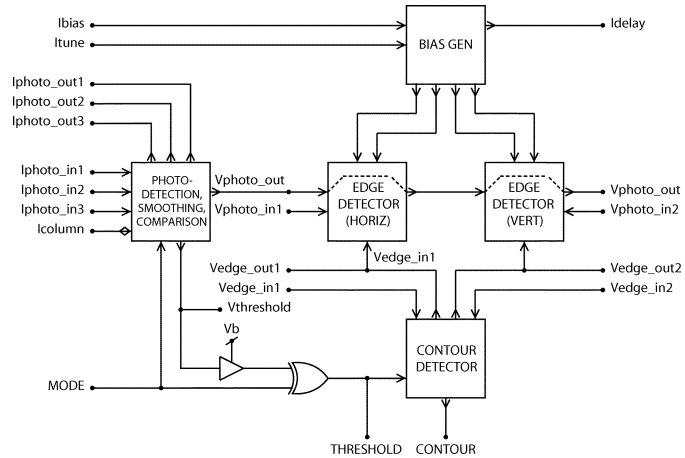


Fig. 6. Schematic diagram of the in-pixel analog signal processing (ASP) organization. Illustrated is the internal connectivity including two edge detector blocks (for horizontal and vertical detection), one photodetection block (for log detection, smoothing and binary thresholding) and a contour detection block (for continuous edge determination).

C. Address Event Representation

As the described system is both asynchronous and data-driven in nature, it is ideally suited to an event-driven output. One such protocol is the Address Event Representation (AER) [20], used extensively in the vision chip arena. The principle behind this data-transmission technique is that each pixel has a unique identifier and upon a pixel generating an event this identifier is asserted onto a digital bus. The data is then communicated off-chip through means of an asynchronous handshake.

IV. Circuit Implementation

This section describes and analyzes the circuits implementing the various sub-blocks, designed for a 0.18-μm CMOS technology. The three following subsections describe the structure and make-up of the three main blocks (ASP, ABP, and AER), the fundamental circuit theory, and the circuit operation. The final subsection presents complete system-level results for scaled processing arrays.

A. Analog Signal Processing

The concept of using a distributed ASP core to feature extract is to avoid the use of analog-to-digital converters (ADCs), reducing complexity and power consumption. Analog processing is used to reduce the computations to a series of comparisons, generating a discrete, asynchronous output. This subsection describes the distributed ASP architecture (see Fig. 6) and circuits for extracting the required CONTOUR and THRESHOLD signals from a matrix of photocurrents.

1) Phototransduction [See Fig. 8(a)]: A parasitic n-well/p-substrate photodiode is used as the phototransductive element (structure and measured characteristics shown in Fig. 7). The structure used includes multiple n-well strips on a p-substrate

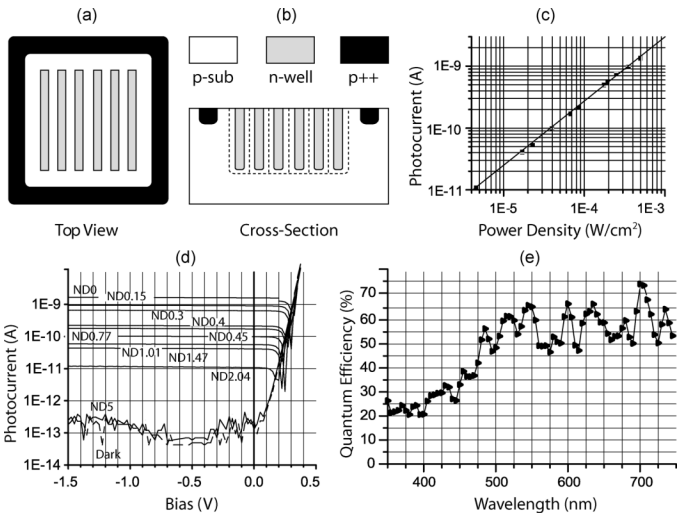


Fig. 7. Details of the n-well/p-substrate photodiode used. Shown are: (a), (b) device structure, (c) responsivity, (d) *IV* characteristics, and (e) spectral quantum efficiency.

connected in parallel within a 28 μm $\times$ 28 μm block. This was chosen in preference to a single well, to maximize the sidewall to base area ratio and boost quantum efficiency; with the added contribution of the deep lateral junctions [21]. This is to overcome reduced quantum efficiencies in vertical junctions in deep submicron CMOS, caused by heavy doping and therefore reduced junction depletion widths.

The photodiode (D1) is reverse-biased by stacking two diode-connected PMOS devices (Q1, Q2) to V_{dd}. The photocurrent range for the given device under the expected light levels is from 100 fA (dark current) to 10 nA. For this current range, devices Q1 and Q2 operate in the weak inversion region and therefore the applied reverse-bias is considered logarithmically proportional to the photocurrent as expressed in (16):

$$V_{\text{photo}} = V_{dd} - (V_{gs1} + V_{gs2}) \approx V_{dd} - 2n\phi_t ln\left(\frac{I_{\text{photo}}}{I_0}\right) \quad (16)$$

where n is the subthreshold slope factor, ϕ_t is the thermal voltage, I_{photo} is the photocurrent, and I_0 is the pre-exponential current.

2) Narrow-Field Local Averaging [See Fig. 8(a)]: The top end diode-connected device is used to form a current mirror with devices Q3–Q6; providing scaled copies of the photocurrent for current-averaging. Devices Q3–Q5 source the adjacent cells with these scaled currents ($I_{\text{photo\_out1,2,3}}$), while device Q6 receives and sums the scaled currents from the adjacent cells to form the four-pixel average, such that

$$I_{\text{local}} = (I_{\text{photo}} + I_{\text{photo\_out1}} + I_{\text{photo\_out2}} + I_{\text{photo\_out3}})/2. \quad (17)$$

3) Wide-Field Local Averaging [Fig. 8(a)]: The wide-field local average is implemented using a column averaging technique. This is facilitated by summing all copied (through current mirror Q7, Q8) narrow-field smoothed currents (I_{local}). Normalization is then achieved by copying this current using a distributed 1:1 current mirror per cell. This has the effect of forming an *X*:1 scaled mirror; with *X* being the number of cells attached to the column.

4) Current-Mode Comparison [Fig. 8(a)]: The near-field local average is then compared to the wide-field column average by means of a basic current comparator formed by an opposing source/sink transistor pair [22] (Q10, Q12). If the device bias points are similar, then both devices will be in saturation and the output voltage ($V_{\text{threshold}}$) is given in (18):

$$V_{\text{threshold}} = \frac{I_{\text{column}}(1 + \lambda_p V_{dd})}{I_{\text{local}}(1 + \lambda_n) + I_{\text{column}}\lambda_p} \quad (18)$$

where λ_n and λ_p are the linear channel length modulation (early) factors for NMOS and PMOS devices, respectively. Subsequently, if the input currents are exactly equal, then this simplifies to

$$V_{\text{threshold}} = \frac{1 + \lambda_p V_{dd}}{1 + (\lambda_n + \lambda_p)}. \quad (19)$$

However, if the source and sink bias points (saturation currents) are different, the device with the higher bias point will be forced out of saturation. For example, in Fig. 8(a), if $I_{\text{column}} > I_{\text{local}}$, the source device (Q12) will operate in the linear region, and the output voltage ($V_{\text{threshold}}$) will swing upwards toward V_{dd}. This behavior is described by (20) and (21):

$$(I_{\text{local}} < I_{\text{column}}),\ V_{\text{threshold}} = \frac{I_{\text{column}}(1 + \lambda_p V_{dd})}{I_{\text{column}}\lambda_p + \frac{I_{\text{local}}}{\phi_t}} \quad (20)$$

$$(I_{\text{local}} > I_{\text{column}}),\ V_{\text{threshold}} = \frac{I_{\text{local}}V_{dd} - I_{\text{column}}\phi_t}{I_{\text{local}} + I_{\text{column}}\lambda_n\phi_t}. \quad (21)$$

Although the generated threshold voltage ($V_{\text{threshold}}$) describes the current comparison discretely for a substantial differential current, smaller current differences cause $V_{\text{threshold}}$ to remain between V_{ss} and V_{dd}. For driving CMOS static logic such a signal is undesirable, as a nondiscrete input can give rise to large "short-circuit" currents. Therefore, a thresholding buffer [see Fig. 8(c)] is required to "square up" this signal to reliable discrete levels.

5) Edge Detection [Fig. 8(b)]: Every pair of adjacent photodiodes are connected to a discrete edge detector [23]. The diode-connected devices [Fig. 8(a)] provide the logarithmically compressed signals (V_{photo1} and V_{photo2}). Two diode-connected devices are stacked in order to ensure the current sourcing device remains in saturation for small photocurrents. The differential voltage ($V_{\text{photo1}}, V_{\text{photo2}}$) is applied to the PMOS differential pair (Q1 and Q2), sourced by a current I_{source}. The drain currents of the differential pair are sunk via current mirror (Q3, Q4, and Q5), which is controlled by the sink current, I_{sink}. The operation is as follows.

I_{source} is selected such that it generates sufficient transconductance to ensure reliable operation for the minimum response time required (limited by response of photodiodes).

I_{sink} is adjusted to lie in between $I_{\text{source}}/2$ and I_{source} and sets the allowable tolerance before indicating an edge and flagging it up. This will set up the gate-source voltages of devices Q4 and Q5 that will determine the maximum current that can be sunk ($I_{d4\max}$ and $I_{d5\max}$, respectively). This circuit operates in one of two states:

- ($V_{\text{photo1}} = V_{\text{photo2}}$): Since $I_{\text{source}}/2 < I_{\text{sink}} < I_{\text{source}}$ then $I_{d1} < I_{d8\max}$ causing device Q4 to be forced into the

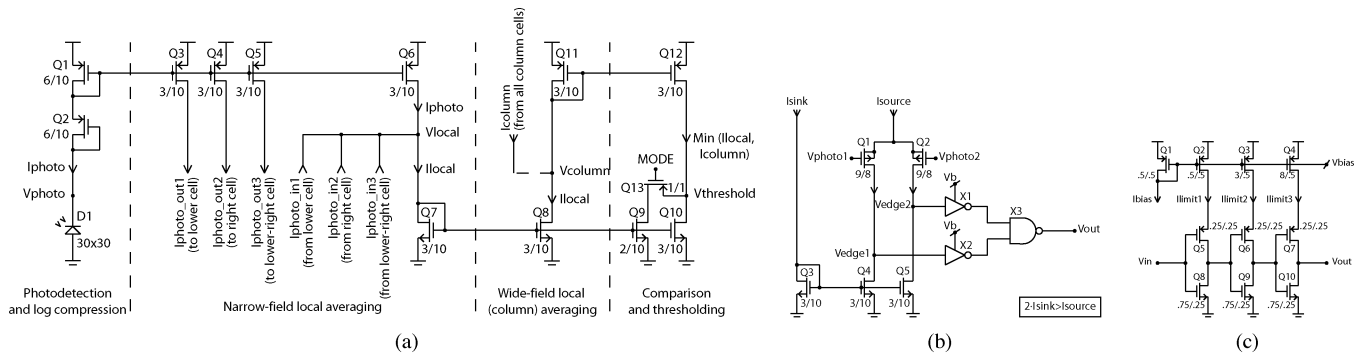


Fig. 8. Front end in-pixel Analog Signal Processing (ASP) circuits for binary feature extraction. Generates the contour/threshold templates feeding the Asynchronous Binary Processing (ABP) core. (a) Photodetection, narrow/wide-field averaging and comparison circuit; (b) tunable edge detector circuit; (c) thresholding inverter.

ohmic region. This in turn will cause V_{edge1} to sit barely above ground and similarly Q5, I_{d6} and V_{edge2} will behave in the same way. As a result of V_{edge1} and V_{edge2} both being low, V_{out} will output high indicating there is no edge.

- ($V_{\text{photo1}} \neq V_{\text{photo2}}$): For example, if $V_{\text{photo1}} < V_{\text{photo2}}$ such that $I_{d1} = I_{d4\max}$ then device Q4 is in saturation and V_{edge1} rises to just below V_{dd}. However, $I_{d2} < I_{d5\max}$ so device Q5 is still in the ohmic region, keeping V_{edge2} low. This will result in V_{out} outputting low, indicating that there is an edge.

Circuit Analysis: Assuming devices Q1 and Q2 are operating in saturation, the following expression (22) can be derived, expressing the output current (differential):

$$I_{d1} - I_{d2} = I_{\text{source}} \cdot tanh\left(\frac{V_{\text{photo1}} - V_{\text{photo2}}}{2n\phi_t}\right) \tag{22}$$

where n is the charge effect due to the substrate (also referred to as the slope factor or subthreshold constant) and ϕ_t is the thermal voltage ($\phi_t = kT/q = 25.9$ mV at room temperature). From (22), the large-signal transconductance of the differential pair can be derived:

$$G_M = \frac{I_{\text{source}}}{2n\phi_t} \cdot sec^2\left(\frac{V_{\text{photo1}} - V_{\text{photo2}}}{2n\phi_t}\right). \tag{23}$$

This can then be used to express the range of values for which the circuit will flag an edge detected.

$$I_{\text{sink}} - [G_M\,(V_{\text{margin}} + |V_{\text{photo1}} - V_{\text{photo2}}|)] < 0 \tag{24}$$

where V_{margin} is the noise margin term expressing the total image noise and device mismatch in the differential pair as an input referred voltage.

This expression directly links into those developed previously for algorithm-based edge detection robustness in (10) and (11). Although (24) yields at least one discrete edge signal $(V_{\text{edge1}}, V_{\text{edge2}})$ for a moderate differential input voltage $(V_{\text{photo1}} - V_{\text{photo2}})$, smaller variations result in graded output levels. As a result, devices Q1–Q5 are in saturation and the edge output voltages are described in (25). As explained previously, thresholding inverters are also inserted in between the edge signals (V_{edge1} and V_{edge2}) and the logic NOR gate to avoid "short-circuit" currents.

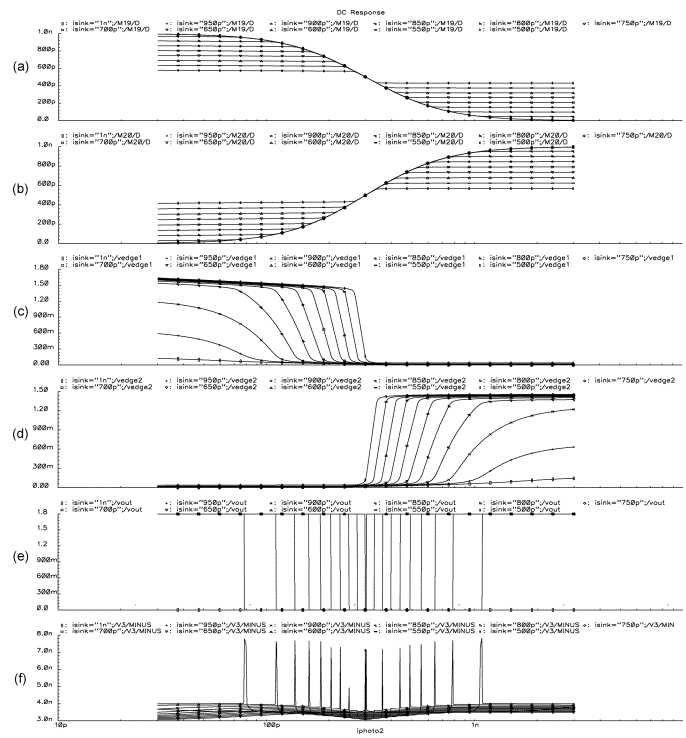


Fig. 9. Simulation results of the edge detector circuit illustrating the tunable sensitivity. Results are for: $I_{\text{source}} = 1$ nA, $500\ \text{pA} \leq I_{\text{sink}} \leq 1$ nA, $I_{\text{photo1}} = 300$ pA, with $1\ \text{pA} \leq I_{\text{photo2}} \leq 10$ nA. Shown (from top to bottom) are: (a) I_{d1}, (b) I_{d2}, (c) V_{edge1}, (d) V_{edge2}, (e) V_{out} and (f) I_{vdd}.

$$V_{\text{edge1,2}} = \frac{1}{\lambda_n}\frac{I_{\text{sink}}}{I_{\text{source}}}\left(1 + e^{\frac{\mp V_{\text{photo2}} \pm V_{\text{photo1}}}{n\phi_t}} - I_{\text{source}}\right) \tag{25}$$

where λ_n is the linear channel length modulation (Early) factor for the sinking devices, assuming $V_{ds} > 3\phi_t$ (for device Q3–Q5). This expression ignores any current source nonidealities (Early effect).

Simulation Results: Fig. 9 illustrates the functionality of this edge detector. This shows the operation of the circuit for a set bias current I_{source} and photocurrent I_{photo1} with varying bias current I_{sink}, with I_{photo2} swept over the entire range (X-axis).

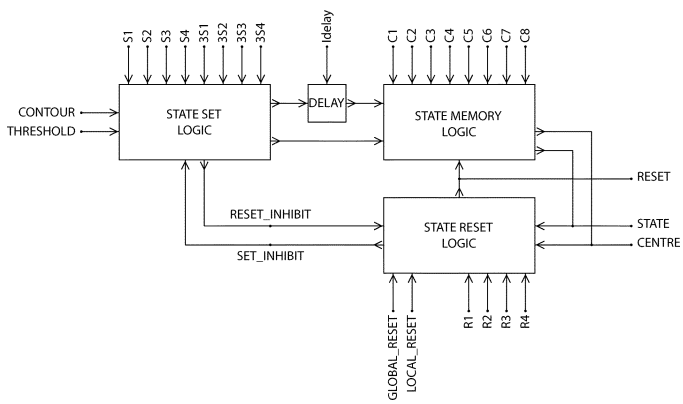


Fig. 10. Schematic diagram of the Asynchronous Binary Processing (ABP) organization. Illustrated is the internal connectivity emphasizing the signal flow path between the various blocks.

This demonstrates the tunability of the edge detection window using a single current input. The corresponding current consumption profile for a 1-nA bias is also given in the simulated results [Fig. 9(f)]. This shows approximately a 3.5-nA average current consumption, peaking to 8 nA at the onset of edge detection.

6) Contour Discrimination: The Boolean expression given in (3) describes the contour discrimination logic present in every cell. This takes its (edge) inputs from the two internal (to that cell) edge detectors (right and bottom edges), the cell to the left (left edge) and the cell above (upper edge).

7) Thresholding [Fig. 8(c)]: To achieve efficient binary extraction, thresholding inverters have been inserted between the digital logic and outputs of the edge and intensity detection functions. This has the task of performing the 1-bit conversion with minimum power consumption. This is implemented using a three-stage cascade of current-starved inverters. For a 1-nA initial stage bias, the optimum current-limit ratio is 1:6:16.

8) Current Generation: All required bias currents ($I_{\rm bias}$, $I_{\rm tune}$, and $I_{\rm aer}$) are generated using a standard proportional to absolute temperature (PTAT) reference [24] using off-chip resistors for versatility and scaled down to lower current values at each stage of duplication in the current distribution scheme.

B. Asynchronous Binary Processing

This section describes the distributed architecture and combinational circuits for facilitating the object segmentation and centroid extraction from a matrix of CENTROID and THRESHOLD binary inputs. The internal architecture of the ABP core is illustrated in Fig. 10. The ABP core consists of three main (functional) blocks for: 1) state setting (inward propagation); 2) state resetting (back-propagation); and 3) state and centroid storage, in addition to some support circuitry.

1) State Set [Fig. 11(a)]: The combinational logic required to set the state generates the SET, SURROUND, and RESET_INHIBIT signals.

- The SET signal is asserted if any adjacent cell has its STATE asserted in addition to its own THRESHOLD being asserted. Alternatively, a CONTOUR signal can assert this signal provided the SET_INHIBIT condition is low.
- The SURROUND signal is required for the centroid detection. This is asserted high if all received surrounding cells have their STATES asserted.
- The RESET_INHIBIT signal is asserted if any adjacent cells have their STATES asserted low.

2) State Reset [Fig. 11(b)]: The combinational logic required to reset the state generates the RESET and SET_INHIBIT signals.

- The RESET signal is asserted when the cell signals a CENTRE (logic X3 and X5). A RESET_INHIBIT signal delays the assertion of the RESET signal until the inward propagation reaches the centroid. This is to guarantee a continuous complete back-propagation path. Alternatively the RESET signal can be asserted if the STATE is asserted and any adjacent cell back-propagates a RESET (logic X4 and X5). On asserting the RESET signal, a monostable is triggered (logic X6, X8–10) to produce a long enough RESET pulse to back-propagate reliably.
- The SET_INHIBIT signal (active low) is asserted if any adjacent cells are resetting. This ensures the back-propagation has reliably terminated before a forward propagation can commence.

3) State Memory [Fig. 11(c)]: The combinational logic required to implement the state memory completes the asynchronous state machine and generates the STATE and CENTRE signals.

- The STATE signal is latched high on assertion of a SET input and conversely it is latched low on assertion of a RESET input.
- The CENTRE signal is asserted when a SURROUND signal is asserted in addition to the cells STATE being asserted low, providing no adjacent cells assert a CENTRE.

4) Discrete Delay [Fig. 11(d)]: An artificial delay is inserted in the SET signal path; between the STATE SET logic and the internal STATE MEMORY cell (Fig. 10). The delay circuit has a binary input (IN) and output (OUT) and two current inputs ($I_{\rm limit}$ and $I_{\rm delay}$). The $I_{\rm limit}$ current defines the current limit on the thresholding inverter circuit. When the input is high, the $I_{\rm delay}$ current is switched (by device Q1) to charge up the capacitor. To generate the output, this is connected to a three stage NMOS current-starved thresholding inverter. Therefore, for a fixed bias, the time delay is given by

$$\tau_{\rm delay} = \frac{Q_c}{I_{\rm delay}} = \frac{C \cdot V_{\rm threshold}}{I_{\rm delay}} \tag{26}$$

where $\tau_{\rm delay}$ is the time delay, Q_c is the charge stored on the capacitor, and $V_{\rm threshold}$ is the threshold voltage of the inverter cascade. For example, an $I_{\rm delay}$ of 1 nA would result in a delay of 102 μs.

C. Address Event Representation

This section outlines the specific AER architecture [25] adopted and the accompanying blocks implemented for off-chip communication. The specific AER architecture implemented is given in Fig. 12, illustrated for a 4×2 array.

Each pixel in the array has a *sender neuron* that latches a pixels state on an event until the data has been transmitted off-

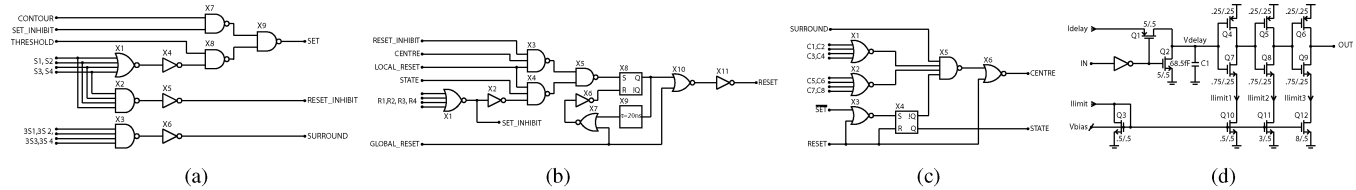


Fig. 11. Back end in-pixel Asynchronous Binary Processing (ABP) circuits facilitating the centroid detection from the contour/threshold binary templates. (a) State set logic. (b) State reset logic. (c) State memory logic. (d) Discrete delay.

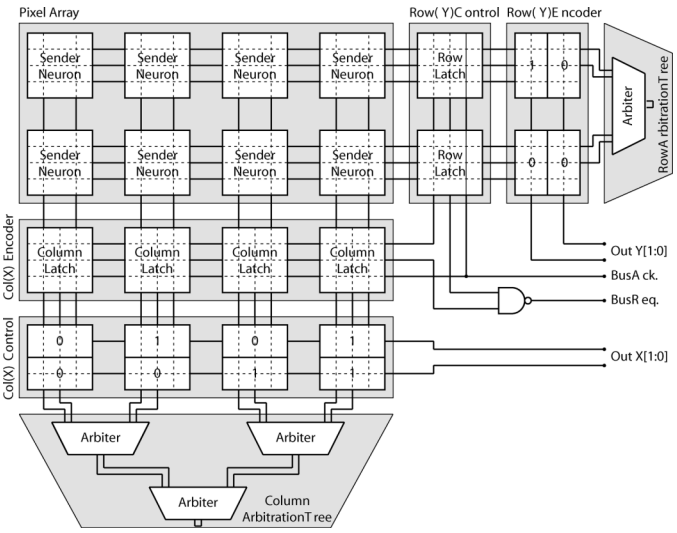


Fig. 12. Address-Event-Representation (AER) architecture for a 4 × 2 array. Illustrated are all required sub-blocks for an AER sending device, excluding pull-up and pull-down biases for shared line drivers.

chip. The sender neuron initially sends an arbitration request to the row *arbitration tree*. The role of the arbitration tree is to select a single output in case of multiply colliding events. On selection of a particular row, the *row header* is latched and subsequently the competition passes to the column arbitration tree. A similar process then occurs from the sender neurons to column arbitration tree and back to the column headers until a single column has been latched. On selection of both a row and column, the chip sends a bus request signal off-chip to the receiving device. The address is read off the bus and then a bus acknowledge signal is relayed back to reset the row and column latches that subsequently reset the sender neuron state. This selection/arbitration process is repeated for all events waiting to be transmitted.

1) Sender Neuron Circuit [Fig. 13(a)]: On a pixel signalling an event, this sequentially requests row and then column arbitration. The sender neuron status is reset only after successful off-chip communication.

2) Column/Row Latch [Fig. 13(b)]: On arbitration, the row or column latch serves to inhibit other requests passing to the arbitration tree until successful off-chip communication.

3) Arbiter Circuit [Fig. 13(c)]: The arbitration tree has the task of selecting one of many requests, facilitated through a binary tree hierarchy. The arbiter cell operates on a single input pair, i.e., by selecting one of two outputs, resolving contention by using a high gain positive feedback element.

TABLE II
COMPARISON BETWEEN EXPECTED AND SIMULATED POWER CONSUMPTION IN CORE COMPONENTS FOR A 12 × 12 ARRAY

Power (nW)	Cellular	Active Cells	Distributed Array	
	Static (Dynamic)	Static (Dynamic)	Static (Dynamic)	Total
ASP	23.89	144	3427	3427
ABP	27.58 (12.21)	144 (50)	3962 (611)	4573
AER	11.16 (31.8)	24 (1)	268 (32)	300
Array (est.)	-	-	7557 (642)	8299
Array (sim.)	-	-	-	8816
Bias (sim.)	32.75	48	-	1572
System (est.)	-	-	-	10388
System (sim.)	-	-	-	9817

4) Address Encoder: The address encoder is based on a simple *wired-OR* topology. As the arbitration tree can only select one input, each output is *hard-wired* to assert the required digital representation on the AER bus.

D. Distributed Simulations

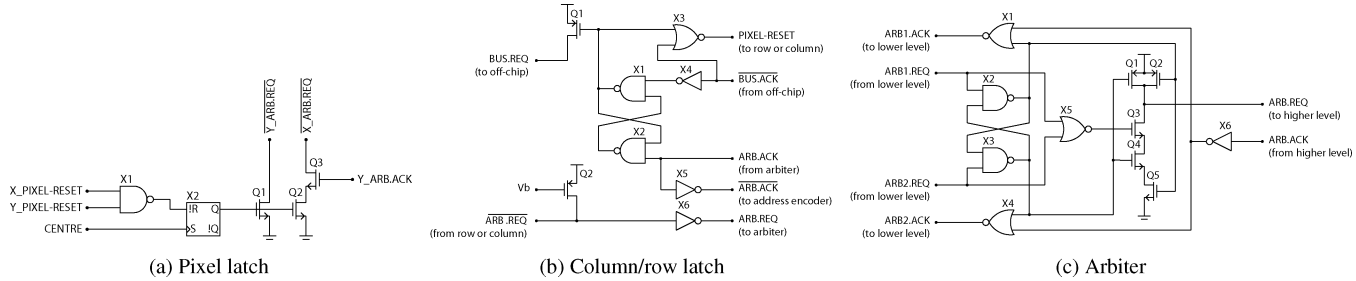
Verification of the distributed architecture is achieved by simulating the individual distributed cores (ASP, ABP and AER) on reduced size arrays.

1) Array: The ABP core is verified by simulating a 9 × 9 array with a static single-object image hardwired; by means of providing the ASP outputs (contour and threshold) as a matrix of distributed inputs within the ABP array. For $I_{\text{delay}} = 1.3$ nA, the transient behavior is illustrated in the simulation results given in Fig. 14. These results illustrate the algorithms *pulsating* action distributed through internal (array) memory for a preset circular object. The cellular STATE can be observed to fill inwards [Fig. 14(a)]. On convergence to a *centroid* cell, the CENTRE signal is flagged, causing a RESET back-propagation [Fig. 14(b)].

2) Power Consumption: Through separate simulations for ASP, ABP, AER, and bias distribution arrays, the constituent power consumptions are summarized in Table II. The simulated results are for 12 × 12 arrays with single object input and typical bias current levels. These results indicate that the static (leakage) dissipation is substantial and in fact is the main source of power consumption within the ABP core; even at high activities.

V. FABRICATED SYSTEM

This section includes details on the fabricated system including the general structure, measured data, and the test procedure.



(a) Pixel latch (b) Column/row latch (c) Arbiter

Fig. 13. Address Event Representation (AER) Circuits for asynchronous off-chip communication. Required are: (a) in-pixel handshake logic for event trigger, (b) column/row header latch for array handshake and (c) arbiter cell (to form binary arbitration tree) for resolution of colliding events. The address encoder used is not shown (a wired-OR topology is used).

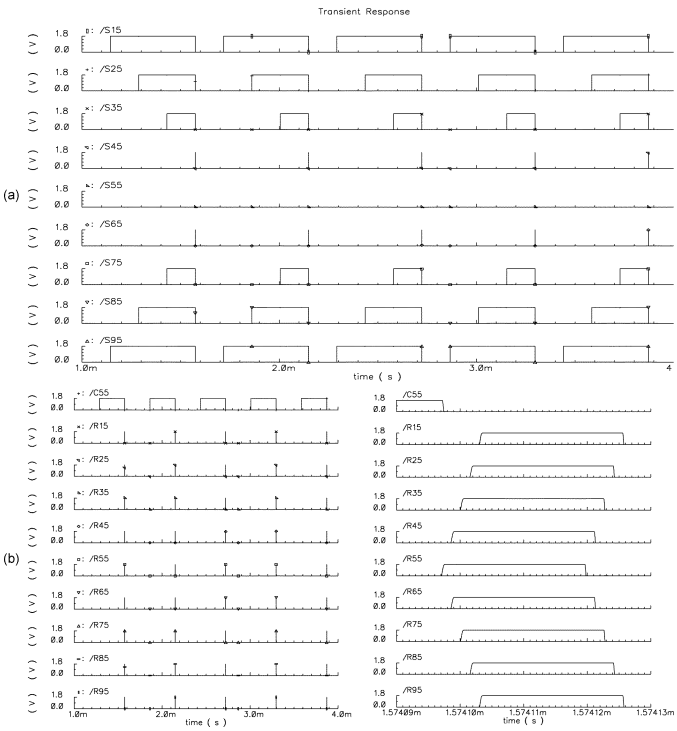


Fig. 14. Transient analysis simulation results for a 9×9 ABP core illustrating bio-pulsating action for a single object image. Results shown are taken across the central row $(Y = 5)$ for: (a) state propagation and (b) reset back-propagation.

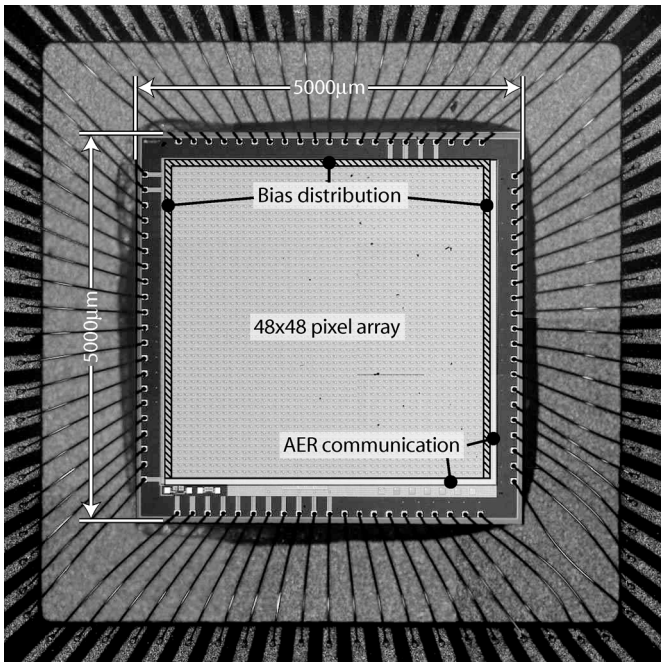


Fig. 15. Microphotograph of the micropower centroiding vision processor.

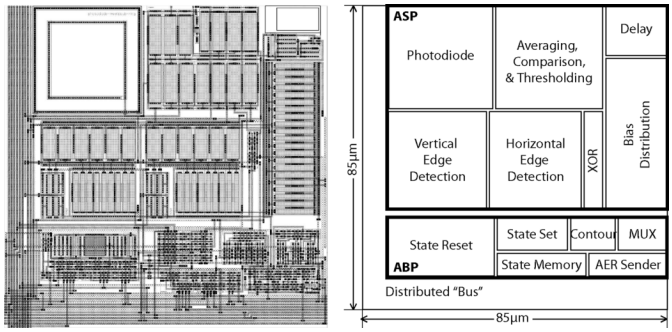


Fig. 16. The regular cell layout (top) and floorplan (bottom). The cell size is $85\ \mu$m $\times$ $85\ \mu$m with $30\ \mu$m $\times$ $30\ \mu$m active photodiode area, giving a 12.5% surface fill factor. Metal layers 5 and 6 have been excluded for clarity.

A. Overview

The complete system is assembled in a 5 mm $\times$ 5 mm die fabricated in a standard 0.18-μm CMOS technology. A microphotograph of the packaged (JLCC84) circuit is shown in Fig. 15. The current distribution circuitry is situated on the left and right of the processing array, with master current references along the top edge. The address event representation hardware is located along the bottom and right edges of the array. This array comprises a 48 $\times$ 48 matrix of cells at 85 μm pitch using a tessellation of nine different cell types (edge, corner and regular) for correct termination to provide full array utilization. The regular cell implementation (layout) is given in Fig. 16.

The pixel floorplan is arranged such that the ASP (approximately top 65% area) is separate from the ABP (approximately bottom 35% area). The distributed data "bus" is routed in metal layers 1 and 3 vertically along the left pixel edge and metal layers 2 and 4 horizontally along the bottom pixel edge. Metal layer 5 is used for current and power supply distribution (horizontal). Metal 6 is used as a ground plane and a light blocking screen, to minimize photo-absorption in the substrate, apart from at photodiode openings.

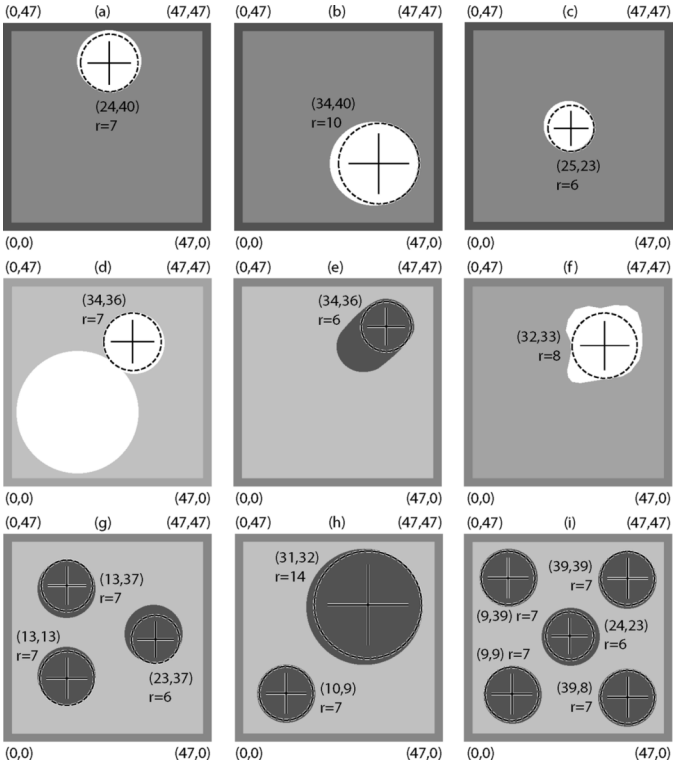


Fig. 17. Test images used to verify system functionality, with results overlayed indicating measured centroid position and size (dotted line representing detected object). Included are: (a–c) regular objects, (d–f) irregular objects, and (g–i) multiple objects.

B. System Verification

A custom testboard has been developed for verifying system functionality. A dedicated microcontroller is used to facilitate the address event handshake and to subsequently store the address event data into internal memory until full, then stream out to a PC via a standard UART (RS232) interface. The limitation of this approach is that the test chip is only tested in short bursts and therefore the output data (although processed in realtime), is only readable off-line. This is due to the limited bandwidth of the UART interface. The full source code for the address-event handshake and sampling is available in [18]. For image acquisition, a 2/3″ format CCTV lens is mounted a fixed distance above the bare silicon surface. Subsequently, a thin-film transistor (TFT) liquid crystal display (LCD) is used to produce the image, positioned perpendicular to the focal plane of the chip.

C. General Functionality

This setup is used to confirm system functionality within the intended design specifications. Sample images, projected onto the chip and corresponding measurements are presented in Fig. 17. This illustrates both single and multiple object detection, centroiding and sizing. Typically the measured centroid and size measurements are within the actual object boundaries, i.e., the system tends to under rather than over estimate. Furthermore, uneven objects are successfully detected but with inaccurate centroid and position estimates, again within the actual object boundaries [see Fig. 17(e), (f)]. However, overlapping objects are erroneously detected as a single uneven object [Fig. 17(d)].

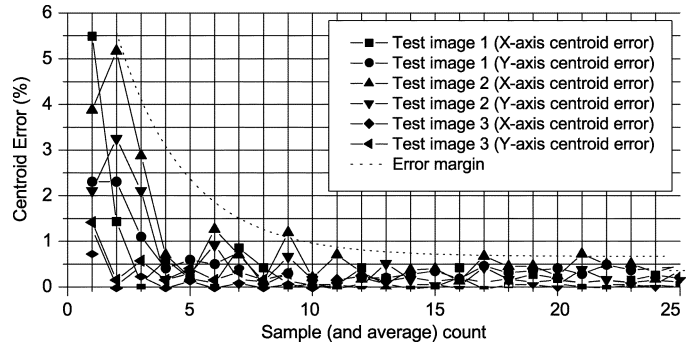


Fig. 18. Pseudo-dithering providing increased centroid position accuracy through successive averaging.

D. Accuracy

The accuracy, as expected, is intrinsically limited to single pixel resolution for both centroid position and object radius. An interesting observation has been a small *random* deviation (± 2 pixels) in object centroid location, resulting in a similar deviation in object size. This is able to provide sub-pixel accuracy (through successive averaging) having a pseudo-dithering effect. This is explained due to an edge effect caused by an imperfectly focused image or a graded object boundary. Subsequently, the static (spatial) fixed-pattern noise (FPN) coupled with the (temporal) flicker noise within the edge detector blocks provide this statistically-biased dithering effect, resulting in a mechanism to enable processing time to be tradable with centroid accuracy. This is illustrated through the trend on measured data shown in Fig. 18.

E. Power Consumption

The measured power consumption levels are generally in line with the previously presented simulated results. The measured results partition the total power consumption into the following sources:

1) Analog Consumption: Representing the ASP power consumption including the photocurrents, local and global averaging and threshold/edge detection circuitry. This is measured to be within 5% of the simulated results, attributed to the fact that all the ASP circuits are biased using current mode techniques. However, the ASP consumption is largely dependant on bias currents and incident light intensity, varying over 15–50 μW, as illustrated in Figs. 19 and 20.

2) Digital Consumption (Leakage): This represents the subthreshold "leakage" current within the digital core. This has been measured to be a main source of power consumption within the ABP core, being in the order of 40–60 μW.

3) Digital Consumption (Static): This represents the static current supply to the digital core. This is virtually entirely due to "digital" short-circuit current caused by incomplete thresholding. This varies depending on the configuration of the edge detection circuitry. This dependance is clearly illustrated in Figs. 19 and 20. ABP static consumption could be expected to account for up to 80% of the total system power requirements in

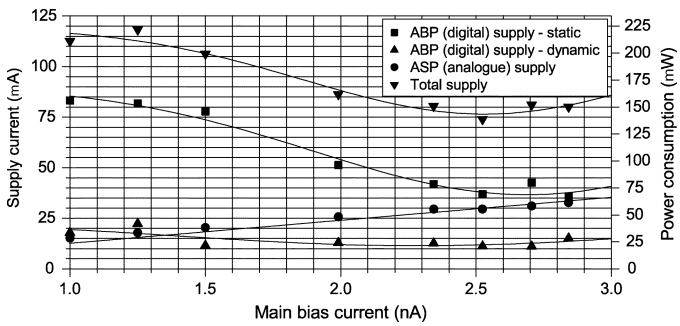


Fig. 19. Measured supply current levels illustrating the effect of tuning main bias current (feeding edge detectors and discrete delays) on system power consumption.

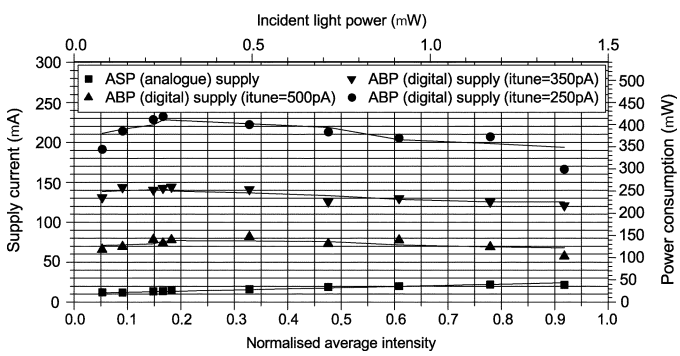


Fig. 20. Measured supply current levels illustrating the effect of illumination level on system power consumption for various tuning bias current levels (controlling the edge detecting threshold).

certain configurations. However, a significantly lower level of static dissipation has been measured from the simulated results. This is thought to be due to the fixed-pattern noise providing a random offset to the edge detector inputs, therefore biasing the differential outputs to always have an offset. As the simulations had only considered images with uniform background intensity, this would represent the maximum static dissipation.

4) Digital Consumption (Dynamic): This represents power consumption directly related to the distributed binary signal propagation and therefore proportional to the activity. In addition the address-event bus activity influences this level. For typical activities, this has been measured to represent only a 10%–15% portion of the total system power consumption. Therefore, no substantial power saving can be achieved by operating the device at a reduced duty cycle, unless the static supply can be modulated.

F. Processing Time

Although the asynchronous nature of this distributed system produces temporally unsynchronised events between different objects (due to the local resetting), the algorithm can be run in a "single-shot" mode, and a clock applied to the global reset input. Using this technique a true high frame rate processor can be realized, the limiting factor being the maximum size of detectable object, i.e., the maximum propagation delay. This is in fact tunable, as the internal propagation delay is a controlled by a bias current, illustrated in Fig. 21. For multiple object processing the "process time" is dictated by the largest sized object.

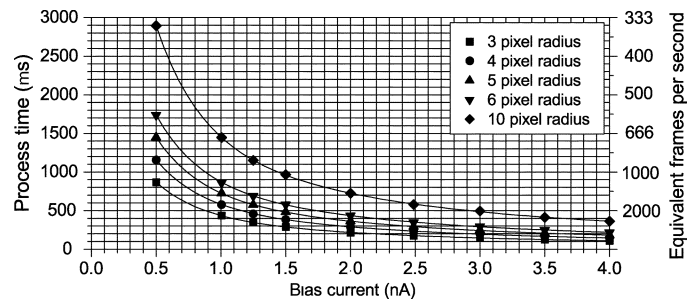


Fig. 21. Dependance of process time on bias current, for input images with maximum object sizes of 3, 4, 5, 6, and 8 pixel radius.

TABLE III
SYSTEM PROPERTIES AND PERFORMANCE SUMMARY

Technology	UMC 0.18μm MM/RF CMOS
Supply voltage	1.8V core (3.3V I/O)
Photosensitivity	$100nW/cm^2$ to $100mW/cm^2$
Responsivity	$0.32A/Wcm^2$ (λ=650nm)
Pixel size	$85\mu m \times 85\mu m$
Surface fill factor	12.46%
Pixel device count	277
Pixel power	96.48 nW (total)
Die dimensions	5mm $\times$ 5mm
Array size	48 $\times$ 48 pixels
System device count	745,200
System power	243.6 μW (total)
Accuracy (centroid and radius)	± 1 pixel
Address-event bandwidth	0.61 MHz (at I_{aer}=1μA)
Image process time	500 μs
Computational capacity	176.52 MIPS
Computational power efficiency	724.64 MIPS/mW
Computational area efficiency	7.06 MIPS/mm^2

VI. DISCUSSION

A focal-plane vision processing chip has been presented for object size and center detection of simple uniform objects. It is the first system reporting multiple (unlimited) object centroid processing capability. Furthermore, the developed system demonstrates high computational efficiency, implementing a computationally intensive algorithm with micropower consumption. Although the developed system includes only a 48 $\times$48 array, with a cellular power budget of a few tens of nanowatts, scaled to a megapixel array this would only require a few tens of milliwatts.

The limitation to scalability would therefore be due to the relatively large pixel footprint. However, this design has not been optimized for silicon area, but for power consumption and reliability. Substantial area savings could therefore be made in three areas: photodiode size, analog device area, and digital optimization. Furthermore, current trends in semiconductor technologies tend to suggest that the next significant advancement will be in 3-D (stacked) CMOS substrates. For distributed systems, this would enable layering the design in true 3-D retinomorphic arrangement [26], therefore achieving reduced footprints and increased resolutions, at increased cost. This would make such systems viable only for specific and demanding applications. For example, the presented system could be most useful in space and military applications, where ultra-low-power and high-speed performance are essential.

The fabricated system has also been shown to utilize fixed pattern noise favorably, both reducing power consumption and increasing accuracy through successive sampling. The achieved system specifications are summarized in Table III.

ACKNOWLEDGMENT

The authors would like to thank T. Sverre Lande and J. Georgiou for many useful discussions and P. Häfliger for providing access to his address-event design libraries.

REFERENCES

[1] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, R. A. Carmona, P. Foldesy, A. Zarandy, P. Szolgay, T. Sziranyi, and T. Roska, "A 0.8 μm CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage," *IEEE J. Solid-State Circuits*, vol. 32, no. 7, pp. 1013–1026, Jul. 1997.

[2] K. A. Boahen, "A retinomorphic vision system," *IEEE Micro*, vol. 16, no. 5, pp. 30–39, Oct. 1996.

[3] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 191–199, Jan. 2003.

[4] A. Fish, D. Akselrod, and O. Yadid-Pecht, "High precision image centroid computation via an adaptive," *Analog Integrated Circuits and Signal Processing*, vol. 39, pp. 251–256, 2004.

[5] B. H. Pui, B. Hayes-Gill, M. Clark, M. Somekh, C. See, J.-F. Piri, S. P. Morgan, and A. Ng, "The design and characterization of an optical VLSI processor for real time centroid detection," *Analog Integrated Circuits and Signal Processing*, vol. 32, no. 1, pp. 67–75, 2002.

[6] N. Massari, L. Gonzo, M. Gottardi, and A. Simoni, "A fast CMOS optical position sensor with high subpixel resolution," *IEEE Trans. Instrum. Meas.*, vol. 53, no. 1, pp. 116–123, Feb. 2004.

[7] R. D. Burns, J. Shah, C. Hong, S. Pepic, J. S. Lee, R. I. Hornsey, and P. Thomas, "Object location and centroiding techniques with CMOS APS," *IEEE Trans. Electron Devices*, vol. 50, no. 12, pp. 2369–2377, Dec. 2003.

[8] J. Akita, A. Watanabe, O. Tooyama, M. Miyama, and M. Yoshimoto, "An image sensor with fast objects' position extraction function," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 184–190, Jan. 2003.

[9] R. A. Blum, C. S. Wilson, P. E. Hasler, and S. P. DeWeerth, "A CMOS imager with real-time frame differencing and centroid computation," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, vol. 3, pp. 329–332.

[10] M. A. Clapp and R. Etienne-Cummings, "A dual pixel-type array for imaging and motion centroin localization," *IEEE Sensors J.*, vol. 2, no. 6, pp. 529–548, Dec. 2002.

[11] G. Erten and S. Hagopian, "Integrated image sensor processor with on-chip centroiding," in *Proc. IEEE Midwest Symp. Circuits Syst.*, 1999, vol. 1, pp. 262–265.

[12] G. Indiveri, "Neuromorphic analog VLSI sensor for visual tracking: circuits and application examples," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 11, pp. 1337–1347, Nov. 1999.

[13] C. S. Wilson, T. G. Morris, and S. P. DeWeerth, "A two-dimensional, object-based analog VLSI visual attention system," in *Proc. Conf. Advanced Research in VLSI*, 1999, pp. 291–308.

[14] T. Horiuchi and E. Niebur, "Conjunction search using a 1-D, analog VLSI-based, attentional search/tracking chip," in *Proc. Conf. Advanced Research in VLSI*, 1998, pp. 276–290.

[15] V. Brajovic and T. Kanade, "Computational sensor for visual tracking with attention," *IEEE J. Solid-State Circuits*, vol. 33, no. 8, pp. 1199–1207, Aug. 1998.

[16] N. M. Yu, T. Shibata, and T. Ohmi, "A real-time center-of-mass tracker circuit implemented by neuron MOS technology," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 4, pp. 495–503, Apr. 1998.

[17] T. G. Constandinou, J. Georgiou, and C. Toumazou, "Toward a bio-inspired mixed-signal retinal processor," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2004, vol. 5, pp. 493–496.

[18] T. G. Constandinou, "Bio-inspired electronics for micropower vision processing," Ph.D. dissertation, Imperial College London, London, U.K., 2005.

[19] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts, "A logarithmic response CMOS image sensor with on-chip calibration," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 1146–1152, Aug. 2000.

[20] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 281–294, Feb. 2004.

[21] J. S. Lee, R. I. Hornsey, and D. Renshaw, "Analysis of CMOS photodiodes. II. Lateral photoresponse," *IEEE Trans. Electron Devices*, vol. 50, no. 5, pp. 1239–1245, May 2003.

[22] C. Toumazou, F. J. Lidgey, and D. G. Haigh, *Analog IC Design: The Current-Mode Approach*. London, U.K.: Peter Perigrinus, 1990.

[23] T. G. Constandinou, J. Georgiou, and C. Toumazou, "Nano-power mixed-signal tunable edge-detection circuit for pixel-level processing in next generation vision systems," *IEE Electron. Lett.*, vol. 39, no. 25, pp. 1774–1775, 2004.

[24] G. Giustolisi, G. Palumbo, M. Criscione, and F. Cutri, "A low-voltage low-power voltage reference based on subthreshold MOSFETs," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 151–154, Jan. 2003.

[25] P. Häfliger, "A spike-based learning rule and its implementation in analog hardware," Ph.D. dissertation, ETH Zürich, , Switzerland, 2000.

[26] J. Georgiou and A. G. Andreou, "A mixed analog/digital asynchronous processor for network models of cortical computation," presented at the 9th Int. Conf. Cognitive and Neural Systems, Boston, MA, 2005.

Timothy G. Constandinou (M'98) received the B.Eng. degree in electrical and electronic engineering with first class honors in 2001 and the Ph.D. degree in electronic engineering from the Imperial College of Science, Technology and Medicine, London, U.K., in 2005.

He is currently with the Institute of Biomedical Engineering at Imperial, where he is the acting Bionics Research Officer. His research interests include biologically inspired vision, biomedical electronics and micropower circuit design.

Dr. Constandinou is a member of the IEEE Circuits and Systems Society, BioCAS Technical Committee.

Christofer Toumazou (M'87–SM'99–F'01) received the Ph.D. degree from Oxford-Brookes University in collaboration with UMIST Manchester, U.K., in 1986.

He is a Professor of Circuit Design in the Department of Electrical and Electronic Engineering, Imperial College London, U.K. His research interests include high-frequency analog integrated circuit design in bipolar, CMOS, and SiGe technology for RF electronics, and low-power electronics for biomedical applications. He has authored or co-authored some 300 publications in the field of analog electronics and is a member of many professional committees. He is founder and Director of the Institute of Biomedical Engineering at Imperial, and the founder and Director of Toumaz Technology Limited.

Prof. Toumazou is a past Chairman for the Analog Signal Processing Committee of the IEEE Circuits and Systems (CAS) Society and past Vice-President of Technical Activities for the IEEE CAS Society. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING and is currently Honorary Editor and Chairman of the IEE *Electronics Letters*. He is co-winner of the IEE 1991 Rayleigh Best Book Award for *Analog IC Design: The Current-Mode Approach*. He is also a recipient of the 1992 IEEE CAS Outstanding Young Author Award for his work on high-speed GaAs op-amp design.