AER Image Filtering Architecture for Vision-Processing Systems

Teresa Serrano-Gotarredona, Andreas G. Andreou, and Bernabé Linares-Barranco

***Abstract*—A VLSI architecture is proposed for the realization of real-time two-dimensional (2-D) image filtering in an address-event-representation (AER) vision system. The architecture is capable of implementing any convolutional kernel $F(x, y)$ as long as it is decomposable into x-axis and y-axis components, i.e., $F(x, y) = H(x)V(y)$, for some rotated coordinate system $\{x, y\}$ and if this product can be approximated safely by a signed minimum operation. The proposed architecture is intended to be used in a complete vision system, known as the boundary contour system and feature contour system (BCS-FCS) vision model, proposed by Grossberg and collaborators. The present paper proposes the architecture, provides a circuit implementation using MOS transistors operated in weak inversion, and shows behavioral simulation results at the system level operation and some electrical simulations.**

***Index Terms*—Analog integrated circuits, communication systems, convolution circuits, Gabor filters, image analysis, image segmentation, neural networks, nonlinear circuits, subthreshold circuits.**

I. Introduction

HUMAN beings have the capability of recognizing objects, figures, and shapes, even if they appear embedded within noise, are partially occluded, or look distorted. To achieve this, the human vision-processing system is structured into a number of massively interconnected neural layers with feedforward and feedback connections among them. Neurons communicate by means of electrical streams of pulses. Each neuron broadcasts its output to a large number of other neurons, which can be inside the same layer or at different layers. The way this is done is through physical connections called *synapses* [1]. One big problem encountered by engineers when it comes to implement bio-inspired (vision) processing systems is to overcome the massive interconnections. An interesting way of trying to solve this is by developing models and algorithms that require a small local interconnectivity among neighboring neurons. Cellular neural networks (CNN's) are one way of doing this [2]–[4]. However, in this paper we will focus on another approach, whose popularity has grown recently, which is known as address even representation (AER) [5]–[8]. Fig. 1 shows a schematic figure, outlining the essence behind AER. Suppose we have an emitter chip containing

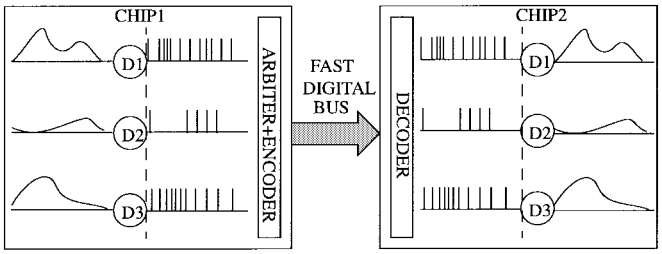


Fig. 1. AER interchip communication scheme.

a large number of neurons or cells D_1, D_2, D_3, ... whose activity changes in time with a relatively slow time constant. For example, if chip one is a retina chip and each neuron's activity represents the illumination sensed by a pixel, the time constant with which this activity changes can be equivalent to the frame rate (i.e., 25–30 changes/s or a time constant of about 30–40 ms)[1]. The purpose of an AER-based communication scheme is to be able to reproduce the time evolution of each neuron's activity inside a second, or receiver, chip, using a fast digital bus with a small number of pins. In the emitter chip, the activity of each pixel has to be transformed into a pulse-stream signal such that pulse width is minimum and the spacing between pulses is reasonably high, to time multiplex the activity of a relatively large number of neurons. Every time a neuron produces a pulse, its address or code should be written on the bus. For the case where more than one pulse is produced simultaneously by several neurons, a classical arbitration tree can be introduced [5]–[7], or one based in winner-takes-all (WTA) row-wise competitions [9], or simply by making no neuron accessing the bus [10]. Whatever method is used, the result will be the presence of a sequence of addresses or codes on the digital bus that one or more receiver chips can read. Each receiver chip must contain a decoding circuitry so that a pulse reaches the neuron (or neurons) which ought to be connected to the emitter chip neuron specified by the address read on the bus. If each neuron integrates the sequence of pulses properly, the original activity of the neurons in the emitter chip will be reproduced. Note that in AER, those neurons that are more active access the bus more frequently. This property allows to optimize the use of the bus, since neurons with low activity will not consume much communication bandwidth. Mathematically, if $I_{\text{in}}(x, y)$ is the intensity (or activity) of pixel at coordinate (x, y) of the emitter chip, this pixel will generate a stream of pulses $\mathcal{P}_o(x, y)$ so

Manuscript received February 24, 1998; revised September 20, 1998. This paper was recommended by Associate Editor T. Roska.

T. Serrrano-Gotarredona and B. Linares-Barranco are with the Instituto de Microelectrónica de Sevilla (IMSE), Centro Nacional de Microelectrónica (CNM), 41012 Sevilla, Spain (e-mail: terese@imse.cnm.es).

A. G. Andreou is with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA.

Publisher Item Identifier S 1057-7122(99)08066-6.

[1] In this paper we consider real time a processing that is performed at a frame rate (30–40 ms) or faster.

that when integrated at the receiver chip, the original activity is recovered

$$\Upsilon \otimes \mathcal{P}_o(x,y) = I_{\text{in}}(x,y). \tag{1}$$

Here, the operator $\Upsilon \otimes$ denotes integration of a sequence of pulses. Usually in AER, this operator is a lossy integration and the asynchronous sequence of pulses $\mathcal{P}_o(x,y)$ is such that its pulse density represents the pixel intensity. The interfacing bus activity is the time-multiplexed sequence of pulses of all active emitter chip pixels $\sum_{x,y} \mathcal{P}_o(x,y)$.

This is the simplest AER-based communication scheme among chips. However, AER allows us to easily add more complicated processing. For example, input images can be translated or rotated by remapping the addresses while they travel from one chip to the next. By properly programming an EEPROM as a look-up table, any address remapping can be implemented by simply inserting the EEPROM between the two chips. Furthermore, many EEPROM's can be connected in parallel, each performing, for example, a rotation at a specific angle and each delivering the remapped addresses to a set of specialized processing chips. It is also possible to include synaptic weighting by having the EEPROM store the weight value, dumping it on a data bus, have the receiver chip read both the address and the data bus, and perform a weighted integration in the destination(s) neuron(s). It is also possible to implement projective fields, i.e., for every address that appears on the bus, a small digital system could generate a sequence of addresses around it and send it to the receiver chip. This would be a time-multiplexed projection-field generation. In the architecture proposed in this paper, we implement a synaptically weighted projection field for each address read on the bus, not in a time-multiplexed manner, but in parallel. This can be done by either having a hard-wired kernel in the filtering chip [11], or by implementing a programmable one, as proposed in this paper.

II. The Programmable Filter

The programmable filter described in this paper is intended to be used in a vision model system, known as the boundary contour system (BCS) and feature contour system (FCS) [12]. Such a vision model consists of an image-sensing layer, followed by a set of illumination normalization layers (this is also known as a retina [7], [8]). The output, which is a contrast image, is applied to a set of orientation-specific edge-extraction Gabor-like filters. Their outputs are then fed to a set of convolutional processing layers, organized in four stages connected in feedback, intended to extract long-range contours of the input image while removing noise. The convolutional kernels used in most of these layers $F(x,y)$ are decomposable into x- and y-axis components $F(x,y) = H(x)V(y)$, for some rotated coordinate system $\{x,y\}$. Using AER allows us to implement a filtering chip only for the coordinate system $\{x,y\}$ for which $F(x,y)$ is decomposable. To do the filtering for another coordinate system $\{x',y'\}$, rotated with respect to $\{x,y\}$, an arbitrary angle α, we can use the same chip, but provide addresses which have been $-\alpha$ rotated previously (by simply inserting an appropriately programmed EEPROM in the interfacing bus).

In the filtering chip, the convolutional kernel is implemented as follows. Every time a pulse for address (p,q) is received, pulses are sent to all pixels in its vicinity. In this way, the lossy integrator at pixel (x,y) of the receiver chip will integrate the sequence of pulses

$$\mathcal{P}_1(x,y) = \sum_{x-L}^{x+L} \sum_{y-L}^{y+L} F(x-p, y-q)\mathcal{P}_o(p,q) \tag{2}$$

which are all pulses coming in from its vicinity, weighted by the convolutional kernel $F(x,y)$. The weighting is performed by modulating the width of each incoming pulse. Thus, every time a pulse is received for pixel (p,q), a pulse of width $F(x-p, y-q)$ is sent to pixel (x,y) in its vicinity ($x \in \{p-L,\ldots,p+L\}, y \in \{q-L,\ldots,q+L\}$). The resulting lossy integral processing of these stream of pulses is the output image

$$\begin{aligned} I_{\text{out}}(x,y) &= \Upsilon \otimes \mathcal{P}_1(x,y) \\ &= \sum_{x-L}^{x+L} \sum_{y-L}^{y+L} F(x-p, y-q)(\Upsilon \otimes \mathcal{P}_o(p,q)) \\ &= \sum_{x-L}^{x+L} \sum_{y-L}^{y+L} F(x-p, y-q) I_{\text{in}}(p,q). \end{aligned} \tag{3}$$

Pulse-width modulation is done as follows. When a pulse for coordinate (p,q) is received, all columns x in the vicinity of column p receive a pulse of width $|H(x-p)|$ and all rows y in the vicinity of rows q receive a pulse of width $|V(y-q)|$. The values of $H(\cdot)$ and $V(\cdot)$ are stored in a small on-chip RAM. The integrator at coordinate (x,y) receives a pulse of width equal to the minimum of $|H(x-p)|$ and $|V(y-q)|$. Consequently, the convolutional kernel the system implemented is an approximation to $F(x,y) = H(x)V(y)$, which is

$$F_m(x,y) = \text{sign}[H(x)]\,\text{sign}[V(y)]\min(|H(x)|, |(V(y)|) \tag{4}$$

the signed minimum of the vertical and horizontal components.

Certainly, replacing a product operation by a signed minimum introduces an error. However, in most bio-inspired vision-processing models, the task performed by a processing layer is mainly of qualitative importance rather than quantitative. Therefore, choosing a certain mathematical kernel or another to perform a given task (such as edge or orientation extraction) should not be too critical for the global operation of a realistic bio-inspired vision model. Nevertheless, whether or not a product can be substituted by a signed minimum should be evaluated for each particular application. In any case, to give a quantitative feeling of the error introduced, Table I shows the resulting normalized square error[2] (NSE) when changing the product by the signed minimum for some typical image processing x- and y-decomposable kernels.

[2]Figure of merit used by Shi [14] to compare different kernels, defined here as $\text{NSE} = \frac{\iint \|F(x,y) - F_m(x,y)\|^2\, dx\, dy}{\iint \|F(x,y)\|^2\, dx\, dy}$.

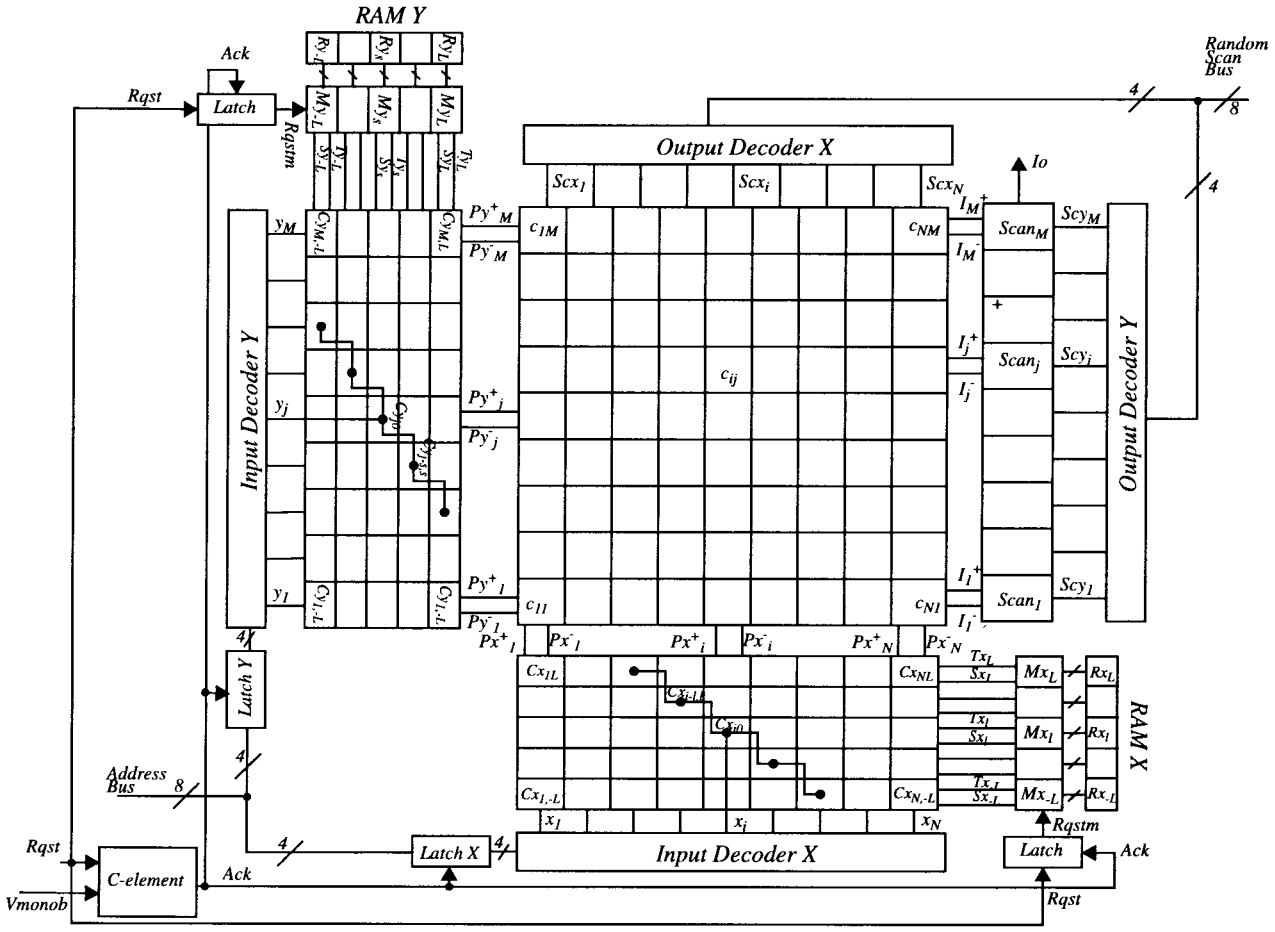


Fig. 2. Floorplan of Complete 2-D filtering system.

III. Circuit Description

This section provides a circuit that implements the previously described functionality. The address bus provides the coordinates (x_0, y_0) of the neuron (or pixel) around which the convolutional kernel should be applied. Pulses will be applied to all rows with a y coordinate in the interval $[y_0 - L, y_0 + L]$ and all columns with an x-coordinate in the interval $[x_0 - L, x_0 + L]$ where $2L + 1$ is the width considered for the kernel. Pulses will be modulated in width, according to function $|V(y)|$ for the rows and function $|H(x)|$ for the columns. At each pixel there is an AND gate, which provides a pulse of width equal to the minimum of $|V(y)|$ and $|H(x)|$. This pulse will generate a fixed-magnitude current pulse of the same width, which will be integrated on a capacitor. Each pixel contains two integrators. One of them, the positive integrator, integrates the pulses of width $\min(|H(x)|, |V(y)|)$ when $\operatorname{sgn}(H(x))\operatorname{sgn}(V(y)) > 0$, while the other, the negative integrator, integrates the pulses when $\operatorname{sgn}(H(x))\operatorname{sgn}(V(y)) < 0$. The values of $V(n)$ and $H(n)$ $(n = -L, \ldots 0 \ldots, L)$ are stored digitally on chip in a small RAM.

Fig. 2 shows the block diagram of the system. It consists of two input decoders that decode the address of the arriving pulse, a C element required for the AER communication protocol [5]–[8], an array of $N \times M$ integrator cells c_{ij}, two sets of programmable monostables $Mx_{-L}, \ldots, Mx_0, \ldots, Mx_L$ and $My_{-L}, \ldots, My_0, \ldots, My_L$ whose pulse widths are controlled by the bits stored in two RAM's, RAMX and RAMY (which store the digital words $Rx_{-L}, \ldots, Rx_0, \ldots, Rx_L$ and $Ry_{-L}, \ldots, Ry_0, \ldots, Ry_L$, respectively), two arrays of $(2L+1) \times N$ and $(2L+1) \times M$ selecting cells $Cx_{i-l,l}$ and $Cy_{j-s,s}$, respectively, two output decoders to select the cells to be scanned, and a scanning circuitry Scan_j to read out an analog output current I_o. Note that in the present prototype of Fig. 2, the system does not generate an AER output. This can be solved by either adding the necessary circuitry to each pixel [5]–[7], which will decrease cell density of the resulting chip, or by adding a postprocessing chip that scans, sequentially, all cells in the array of Fig. 2 and generates an AER output. Once the filter has an AER output, an arbitrary number of filtering stages can be cascaded.

The operation of the system in Fig. 2 is as follows. In RAMX and RAMY digital words of $n+1$ bits are stored $(Rx_{-L}, \ldots, Rx_l, \ldots, Rx_L$ and $Ry_{-L}, \ldots, Ry_s, \ldots, Ry_L)$. The first bit Sx_l (or Sy_s) indicates the sign of the function $H(x)$ (or $V(y)$). The following n bits indicate the absolute value $|H(x)|$ (or $|V(y)|$). These n bits linearly control the length of the pulse triggered by monostables Mx_l (or My_s). The monostables achieve this by charging with a constant current a programmable capacitor controlled by the n bits in Rx_l or Ry_s. Hspice simulations showed a linear relationship between digital code word and pulse width. The pulses generated by the monostables are sent through lines Tx_l (or

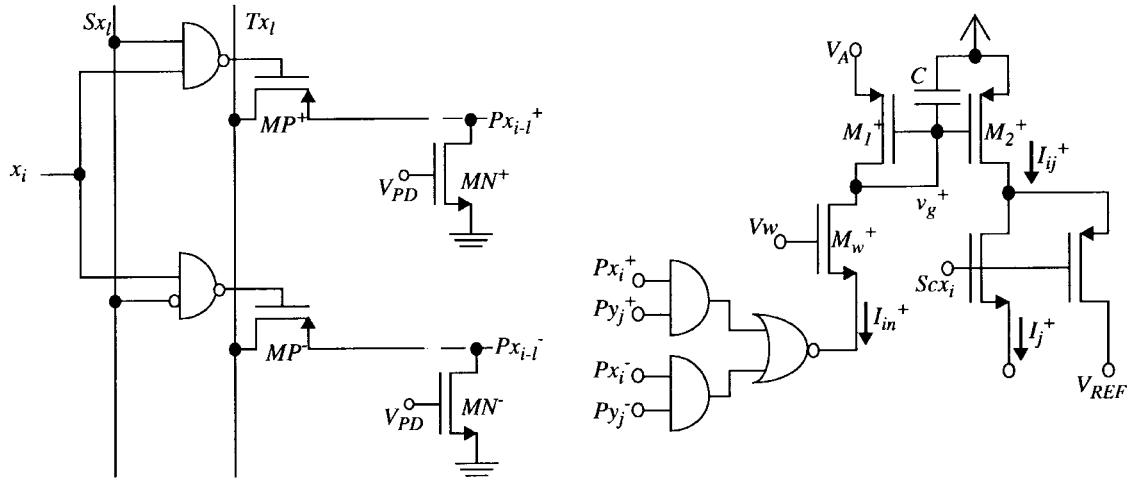


Fig. 3. Schematic of (a) the neighborhood-selection cell and (b) one half of the core-cell diode-capacitor integrator.

Ty_s) and are triggered whenever an external pulse arrives to the system. When an external pulse arrives, the input decoders activate lines x_i and y_j, corresponding to the address of the arriving pulse. The selection cells controlled by x_i (cells $Cx_{i-l,l}$ in Fig. 2, $l = -L, \ldots 0 \ldots, L$) connect the pulse in line Tx_l to line Px^+_{i-l} if the sign bit Sx_l is one. If the sign bit Sx_l is zero, line Tx_l is connected to the negative line Px^-_{i-l}. In this way, pulses Tx_l (or Ty_s) are sent through lines Px^+_{i-l} or Px^-_{i-l} (Py^+_{j-s} or Py^-_{j-s}), depending on the sign of the weight stored in Rx_l (or Ry_s).

Each neuron c_{ij} has two integrators. The positive integrator accumulates charge when pulses are simultaneously arriving through horizontal and vertical lines of the same sign. That is, it integrates a pulse when lines Px_i^+ and Py_j^+ (or lines Px_i^- and Py_j^-) are simultaneously high or, equivalently, it performs the operation $(Px_i^+ \cap Py_j^+) \cup (Px_i^- \cap Py_j^-)$. Hence, the positive integrator in cell c_{ij} computes along time the following sum:

$$I^+(x_i, y_j) = I_w \sum_{\substack{p,q \\ \mathrm{sgn}(H)\mathrm{sgn}(V)>0,}} \min(|H(x_i - p)|, |V(y_j - q)|) \times (\Upsilon \otimes \mathcal{P}_o(p,q)) \quad (5)$$

where $\Upsilon \otimes \mathcal{P}_o(p,q)$ is the (lossy) integral over time of the number of pulses the address bus receives for pixel (p,q) and I_w is the fixed magnitude of the current pulses being integrated. Similarly, the negative integrator accumulates charge when pulses arriving through horizontal and vertical lines of opposite sign Px_i^+ and Py_j^- (or Px_i^- and Py_j^+) are simultaneously high, that is, it performs the operation $(Px_i^+ \cap Py_j^-) \cup (Px_i^- \cap Py_j^+)$. Consequently, the difference between the outputs of the positive and negative integrators is given by

$$I_w \sum_{p,q} \mathrm{sgn}(H(x_i - p))\,\mathrm{sgn}(V(y_j - q)) \times \min[|H(x_i - p)|, |V(y_j - q)|](\Upsilon \otimes \mathcal{P}_o(p,q)) \quad (6)$$

which is the filter operation we want to implement.

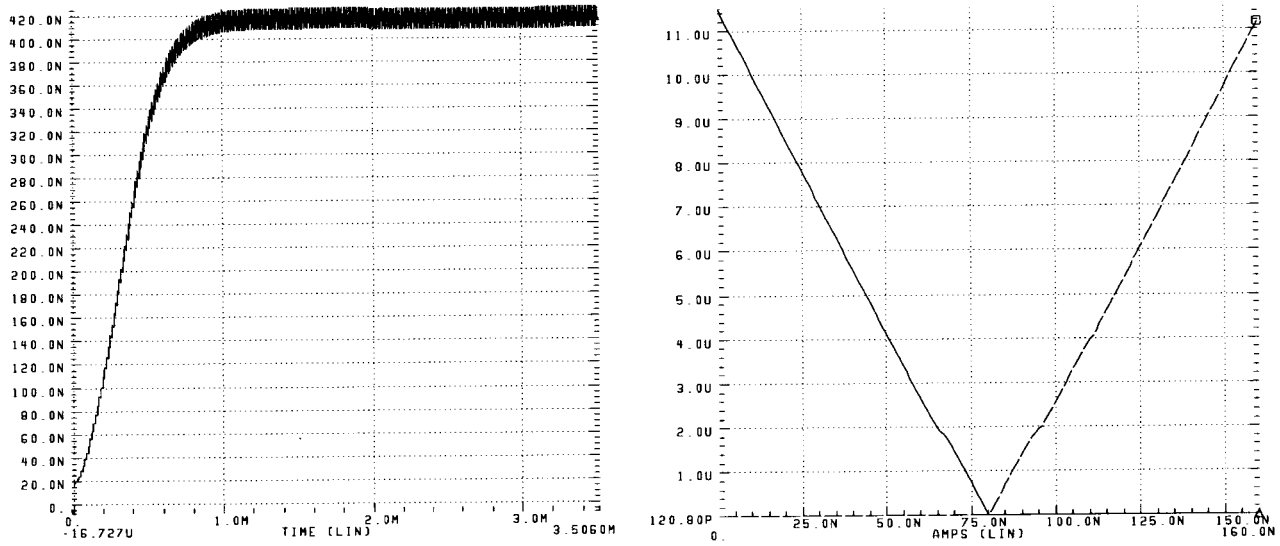
Fig. 3(a) depicts the schematic of the selection cell $Cx_{i-l,l}$ (or $Cy_{i-s,s}$) used to select the neighborhood of cells where the monostable pulses have to be sent. It consists of two NAND gates controlling the PMOS switches MP^+ and MP^- and two NMOS pull-down transistors MN^+ and MN^-. Each selection cell $Cx_{i-l,l}$ has two control signals (the decoder output x_i and the sign bit Sx_l from RAMX), one input signal (the monostable output Tx_l), and two outputs (Px^+_{i-l} and Px^-_{i-l}). When a pulse arrives with address (x_i, y_j), it activates the decoders output x_i and y_j, respectively. The decoder output x_i controls all the selection cells $Cx_{i-l,l}$ with $l \in [-L, \ldots, L]$. When x_i is high, if the sign bit Sx_l is one, the selection cell $Cx_{i-l,l}$ connects the monostable output line Tx_l to the positive line Px^+_{i-l}. If the sign bit Sx_l is zero, line Tx_l is connected to the negative line Px^-_{i-l}. The same is valid for the Y-coordinate selection cells.

Each synaptic cell, c_{ij} has two integrators: the positive and the negative. Fig. 3(b) shows the circuit diagram for the positive integrator. The negative is identical, except for labeling. The integrator is based on the capacitor–diode integrator concept for subthreshold MOS operation [7]. As will be seen next, this integrator has some interesting properties with respect to a conventional linear RC-integrator.

- The steady-state current is proportional to pulse stream frequency.
- The steady-state current is proportional to pulse width
- The steady-state current ripple is independent of the current level.

In Fig. 3(b), the two AND and the NOR gates provide a pulse of width equal to the minimum of the pulse width coming in horizontally and vertically. This pulse turns ON current source M_w, providing a current pulse of amplitude I_w (controlled by bias voltage V_w). Since transistors M_1^+ and M_2^+ are biased in subthreshold, the integrator input and output currents I^+_{in} and I^+_{ij} are related by [7]

$$Q_T \frac{dI^+_{ij}}{dt} = I^+_{ij}\left(I^+_{\mathrm{in}} - \frac{I^+_{ij}}{A}\right) \quad (7)$$



(a) (b)

Fig. 4. Hspice simulation of (a) integrator cell transient and (b) dc characteristics of scan-out cell.

where $A = \exp((V_{\rm dd} - V_A)/\nu_T)$, $Q_T = C\nu_T/\kappa$, ν_T is thermal voltage and κ is a characteristic subthreshold dimensionless technology parameter, whose value may range from 0.60 to 0.98 [14]. When a train of pulses of width T_h and frequency $1/T$ ($T \gg T_h$) is applied to this integrator, the steady-state output current is [7]

$$I_{ij}^{+}(\infty) \approx A I_w \frac{T_h}{T} \tag{8}$$

with a ripple of

$$\frac{\Delta I_{ij}^{+}(\infty)}{I_{ij}^{+}(\infty)} \approx \frac{T_h}{\tau} \tag{9}$$

where $\tau = Q_T/I_w \gg T_h$. Equation (9) shows that the relative resolution in the integrator output is constant, independent of the signal level. According to (8), each integrator outputs a current which is proportional to the frequency $1/T$ and width T_h of the input pulses. If the AER input-image pixel intensity is linearly encoded with the frequency of the arriving pulses and the convolutional kernel is encoded as the pulses width, the output current of the positive integrators would be the input image, filtered with the filter positive terms. Equivalently, the negative integrator output current would be the input image, filtered with the negative terms of the filter. Hence, the result of subtracting the output current of the negative integrator from the output current of the positive one is the filter output.

Fig. 4(a) shows an Hspice transient simulation for one of the integrator cells in Fig. 3(b). Transistor sizes are $W = 12$ μm and $L = 12$ μm, the integrating capacitor is $C = 0.1$ pF, pulse amplitude is $I_w = 13.5$ nA, pulse width is $T_h = 100$ ns, frequency of pulse stream is $1/T = 80$ KHz, $V_{\rm dd} = 5$ V, and voltage V_A was set to 4.67 V (which yields a current gain from transistor $M_1^{\pm}$ to $M_2^{\pm}$ of around 2000). Similar simulations were performed by sweeping the frequency of the input pulse stream $1/T$ and the width of the pulses T_h. The results are shown in Fig. 5. Fig. 5(a) shows the steady-state current level as a function of frequency, while maintaining $T_h = 10$ ns. Fig. 5(b) shows the steady-state current level as a function of pulse width, while maintaining the frequency constant at 4 KHz.

Sometimes in 2-D image-filtering processing a rectification operation has to be performed. This is the case, for instance, when doing orientation extraction with Gabor-like kernel filters. The output of the filter is rectified for each pixel [12]. Because of this, the chip scan-out circuitry, which brings out of the chip the state of a c_{ij} cell, has been designed to be able to add a rectification operation. The random-access scanning circuitry can read the rectified output current of any cell selected by the random scan bus of Fig. 2. The output decoder X (see Fig. 2) selects a column i through line Scx_i. When a column is not selected, the output currents I_{ij}^{+} and I_{ij}^{-} of all cells c_{ij} in that column flow to a line of constant voltage $V_{\rm REF}$ [see Fig. 3(b)]. If column i is selected, currents I_{ij}^{+} and I_{ij}^{-} of all cells c_{ij} in these columns flow to lines I_j^{+} and I_j^{-}, respectively, of the scan-out cell Scan_j, shown in Fig. 6. Each scan-out cell Scan_j receives two input currents I_j^{+}, I_j^{-} and provides an output current $|I_j^{+} - I_j^{-}|$. Current I_j^{-} is mirrored through a PMOS current mirror and subtracted from current I_j^{+}. The PMOS current mirror has an active input [15], clamped to a voltage $V_{\rm REF}$. This maintains a constant voltage $V_{\rm REF}$ at output nodes $I_{ij}^{\pm}$ of cells c_{ij} when they are selected, thus speeding up the read out of currents. Current $I_j^{+} - I_j^{-}$ enters the current comparator composed of transistors M_1, M_2 and OPAMP_1 [16], whose input node (and output I_{ij}^{+} of all selected c_{ij} cells) is clamped to voltage $V_{\rm REF}$. If current $I_j^{-} - I_j^{+}$ is positive, transistor M_2 will sink this current. Transistor M_4 shares its gate with M_2 and its source is connected to a voltage reference of value $V'_{\rm REF}$, thus, transistor M_4 mirrors the current passing through M_2,

$$[I_j^{-} - I_j^{+}]^{+} = \begin{cases} I_j^{-} - I_j^{+} & \text{if } I_j^{-} - I_j^{+} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The precision of this current reflection depends on how tightly the source of M_2 is clamped to voltage $V_{\rm REF}$. To achieve a

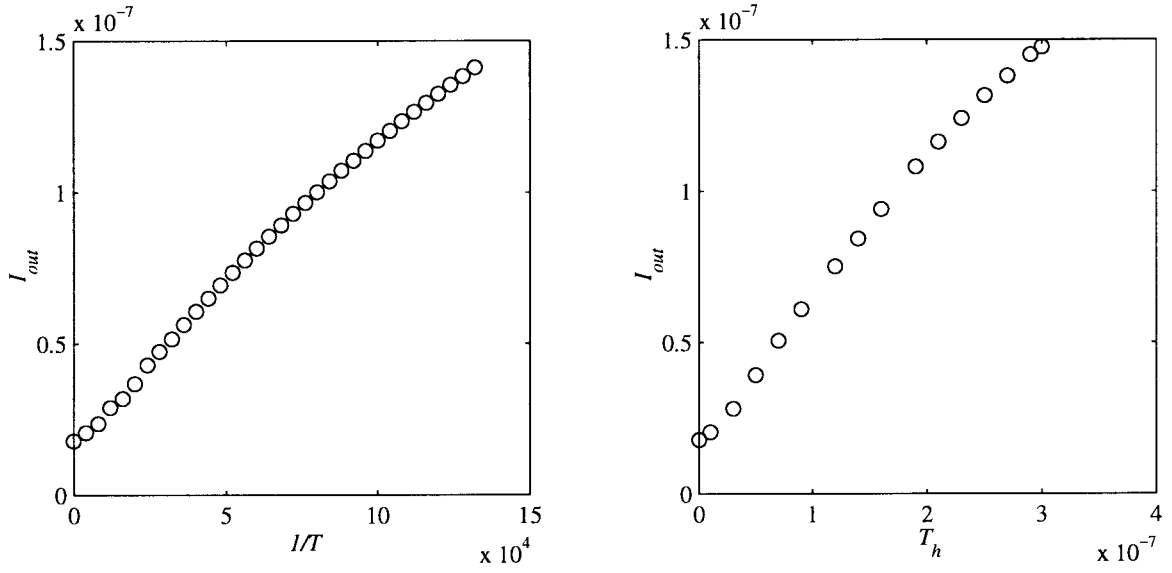


Fig. 5. Hspice simulation results of core-cell integrator. (a) Steady-State current versus frequency of input pulse stream for $T_h = 10$ ns. (b) Steady-State current versus T_h for $1/T = 4$ KHz.

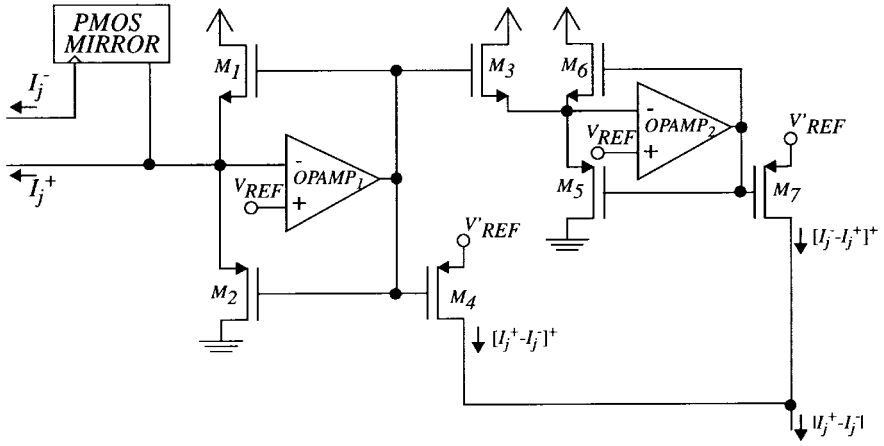


Fig. 6. Schematic of a cell to scan out the absolute value of the difference of two currents.

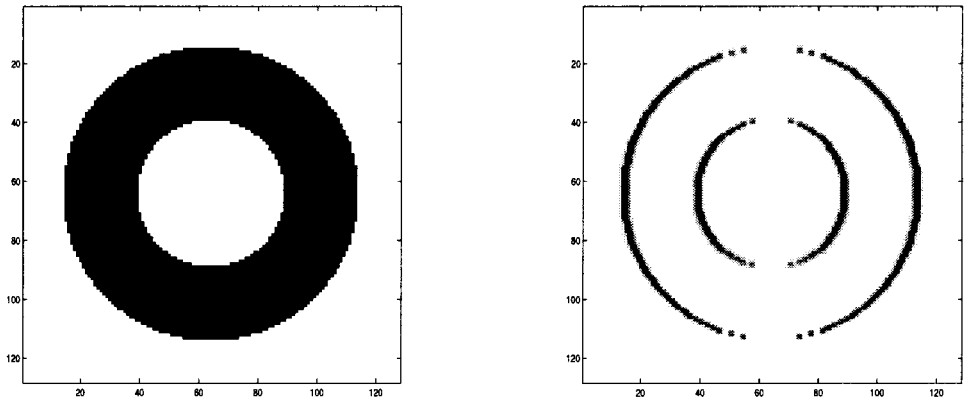
good precision a high gain opamp is needed, although this would slow down the process. Thus, a compromise between speed and precision must be taken. If current $I_j^+ - I_j^-$ is positive, transistor M_1 sources this current, which is mirrored by transistor M_3 because its source is clamped to V_{REF} by the current comparator composed of transistors M_5, M_6 and OPAMP$_2$. Therefore, the current through M_3 and M_5 is

$$[I_j^+ - I_j^-]^+ = \begin{cases} I_j^+ - I_j^- & \text{if } I_j^+ - I_j^- > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

This current is again reflected by the PMOS transistor pair M_5, M_7. At the output node, the currents through transistors M_4 and M_7 are added together to get the rectified current $I_o = |I_j^+ - I_j^-|$. Since transistors $M_1 - M_7$ operate in weak inversion, increasing the source voltage of transistors M_4 and M_7, with respect to V_{REF}, will make the current mirrors M_2 M_4 and M_5, M_7 to have a gain higher than one (actually, the gain will be exponentially controlled by this voltage difference). This allows us to have a current gain such that the output current is of the order of hundreds of μA or even some mili-A, making it possible to drive this current directly off-chip at high speeds. Fig. 4(b) shows an Hspice simulation of the dc characteristic of a scan cell. In this simulation, current I_j^+ was set to 80 nA and current I_j^- was swept from 0 nA to 160 nA. Two traces are shown in Fig. 4(b). The dotted line shows the current $[I_j^- - I_j^+]^+$ flowing through transistor M_4. The solid line corresponds to current $[I_j^+ - I_j^-]^+$ flowing through transistor M_7.

IV. Timing Considerations

The time response of the programmable 2-D image filter chip is dominated by the settling time of the integrators in cells c_{ij} when they are fed by pulse streams. For the simulation of Fig. 4(a), for example, the settling time is about 1 ms. However, a general analysis would be as follows. For the diode–capacitor integrator, fed with a stream of pulses of width T_h and frequency $1/T$, it is easy to find that the current through



(a) (b)

Fig. 7. System-Level behavioral simulations of a 128 × 128 array. (a) Input image. (b) Output Image of pseudo-gabor filter extracting vertical edges.

the diode before two consecutive pulses are related by [17]

$$\frac{1}{I_{ij}^{+}((n)T)} = \frac{e^{-\frac{T_h}{\tau}}}{I_{ij}^{+}(n-1)(T)} + \frac{1}{AI_w}\left\{\frac{T-T_h}{\tau} + 1 - e^{-\frac{T_h}{\tau}}\right\}. \tag{12}$$

Calling $x(n) = 1/I_{ij}^{+}(nT)$, $R = e^{-\frac{T_h}{\tau}}$, and S the second term of the right-hand side in (12), this equation can be rewritten as

$$x(n) = x(n-1)R + S = x(0)R^n + S\frac{1-R^n}{1-R} \tag{13}$$

which converges to $x(\infty) = S/(1-R)$, as anticipated by (8). Consequently, (13) can be rewritten as

$$x(n) = x(0)R^n + x(\infty)(1-R^n). \tag{14}$$

The number of pulses n_s required to reach the steady state with a relative error p is given by the solution of[3] $x(n_s) = x(\infty)(1\pm p)$, which yields $R^{n_s} = p/|1-x(0)/x(\infty)|$ or, equivalently,

$$n_s = \frac{\tau}{T_h}\ln\left|\frac{1-\frac{x(0)}{x(\infty)}}{p}\right| \tag{15}$$

The maximum pulse stream frequency $f_{\max} = 1/T_{\min}$ is limited by the communication throughput we would like to achieve. For example, consider the case of having a 128 × 128-pixel contrast output retina [7], [8] (i.e., the output image is already normalized with respect to contrast). Suppose also that, for a real-world image, the average contrast level is as if 10% of the pixels were maximum and the rest minimum. In this case, we would need to allocate, on the average, 128 ×128 × 0.1 pulses of width T_h in a time equal to $T_{\min} = 1/f_{\max}$. A reasonable value for T_h that would assure a good range for the pulse-width modulation described in Section IV could be $T_h = 100$ ns. This would yield $T_{\min} = 164$ μs or $f_{\max} = 6.1$ KHz. Consequently, maximum pixel activity should be coded with $f_{\max} = 6.1$ KHz.

[3]The negative sign is used when $x(0) < x(\infty)$ and the positive when $x(0) > x(\infty)$.

TABLE I
NORMALIZED SQUARE ERROR

kernel	$F(x,y) = H(x)V(y)$	parameters	NSE (dB)
Gaussian	$e^{-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2} e^{-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2}$	$\sigma_x = 10$ $\sigma_y = 15$	-24.92
Even Gabor	$e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} e^{-\frac{1}{2}\left(\frac{y}{\sigma}\right)^2} \sin\left(2\pi\frac{y}{y_s}\right)$	$\sigma = 15$ $y_s = 20$	-19.04
Odd Gabor	$e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} e^{-\frac{1}{2}\left(\frac{y}{\sigma}\right)^2} \cos\left(2\pi\frac{y}{y_s}\right)$	$\sigma = 15$ $y_s = 20$	-19.03
Displaced Gaussians	$e^{-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2}\left(e^{-\frac{1}{2}\left(\frac{y-y_s}{\sigma_y}\right)^2} - e^{-\frac{1}{2}\left(\frac{y+y_s}{\sigma_y}\right)^2}\right)$	$\sigma_x = 15$ $\sigma_y = 5$ $y_s = 5$	-22.73

V. SYSTEM LEVEL OPERATION BEHAVIORAL SIMULATIONS

Up until this point, electrical (Hspice) simulations of some of the circuit components have been presented. However, to validate the functionality of the proposed architecture, some system level (behavioral) simulations are mandatory. In this section we provide such simulations, using MATLAB on the architecture of Fig. 2 for a system of 128 × 128 cells. The input image fed to the system is shown in Fig. 7(a), and the programmed convolutional kernel was a displaced Gaussian (see Table I). Using MATLAB, the AER stream of addresses that this image could generate was computed. The stream of pulses flowing through the bus is characterized by a sequence $(x_i(t_n), y_j(t_n), t_n)$ $n = 0, 1, 2, \ldots$ where $(x_i(t_n), y_j(t_n))$ is the address present on the bus at time t_n. This stream of addresses was then used to control the mathematical model of the architecture of Fig. 2. Each one of the 128 × 128 cells c_{ij} is characterized by the state of two integrators: the positive integrator I_{ij}^{+} and the negative one I_{ij}^{-}. The state of the integrators is controlled by the following differential equations [see (7)]

$$\begin{aligned} Q_T\frac{dI_{ij}^{+}}{dt} &= I_{ij}^{+}\left(I_{\text{in}}^{+} - \frac{1}{A}I_{ij}^{+}\right) \\ Q_T\frac{dI_{ij}^{-}}{dt} &= I_{ij}^{-}\left(I_{\text{in}}^{-} - \frac{1}{A}I_{ij}^{-}\right) \end{aligned} \tag{16}$$

whose solutions were computed analytically. These solutions were used to update the state of the integrators in the following manner. For each address $(x_i(t_n), y_j(t_n))$ present on the bus, all cells c_{pq} in the range $\{p \in [i-L, i+L], q \in [j-L, j+L]\}$ were accessed. For each accessed cell, the pulse width $T_h(p,q)$ was computed, using the approximation of (4) and the simulation results for the monostable. Depending on the resulting sign, either the positive or the negative integrator was updated. After an integrator has been updated, the present time was stored for it so that the next time it needs to be updated, the simulator can compute properly its discharge amount. For each cell c_{ij}, its output is given by $|I_{ij}^{+} - I_{ij}^{-}|$. Using this method until all integrators have reached their steady state within 1% tolerance, results in the system output depicted in Fig. 7(b). In this case, addresses were not prerotated, so that the system is extracting vertical edges. As can be seen, pixels around vertical edges result in a very high output value, while as the edge angle around a pixel deviates from vertical, its output value smoothly decreases until zero.

VI. Conclusion and Future Work

An architecture that implements a programmable 2-D image filter has been presented. The architecture allows us to implement any 2-D filter $F(p,q)$, decomposable into x-axis and y-axis components $F(p,q) = H(p)V(q)$ such that the product can be approximated by a signed minimum. Positive and negative values of $H(p)$ and $V(q)$ can be programmed. The architecture requires an AER input. This allows us to rotate the 2-D convolution kernel to any angle.

A VLSI circuit implementation that realizes the proposed architecture is provided. Circuit simulation results of critical components were given. System-level behavioral simulations of a 128 $\times$ 128 array have been included, which validate the proposed approach. Cell size is 67.2 μm$\times$ 72.6 μm if no AER output is available and 75 μm$\times$ 90.6 μm if AER output is included, for a 1.2-μm double-poly double-metal CMOS process. This would allow, for a 1-cm^2 die, to implement a 2-D filter with approximately 128 $\times$ 128 pixels for no AER output, and 120 $\times$ 100 pixels if AER output is provided. Future work includes the fabrication of a test prototype, testing it with a retina chip with AER output, and assembling a cascade of convolutional processing layers to implement a vision-model system.

References

[1] G. M. Shepherd, *The Synaptic Organization of the Brain*. London, U.K.: Oxford Univ. Press, 3rd Ed., 1990.

[2] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

[3] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273–1290, Oct. 1988.

[4] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Trans. Circuits Syst., II*, vol. 40, pp. 163–173, Mar. 1993.

[5] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Trans. Neural Networks*, vol. 4, pp. 523–528, May 1993.

[6] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*. Amsterdam, The Netherlands: Kluwer, 1994.

[7] K. Boahen, "Retinomorphic vision systems," in *Proc. Microneuro'96: Fifth Int. Conf. Neural Networks Fuzzy Systems*, Lausanne, Switzerland, Feb. 1996.

[8] A. G. Andreou, R. C. Meitzler, K. Strohbehn, and K. A. Boahen, "Analog VLSI neuromorphic image acquisition and pre-processing systems," *Neural Networks*, vol. 8, nos. 7/8, pp. 1323–1347, 1995.

[9] Z. Kalayjian, J. Waskiewicz, D. Yochelson, and A. G. Andreou, "Asynchronous sampling of 2D arrays using winner-takes-all arbitration," in *Proc. 1996 IEEE Int. Symp. Circuits Syst. (ISCAS'96)*, Atlanta, GA, 1996, vol. 3, pp. 393–396.

[10] A. Mortara, E. A. Vittoz, and P. Venier, "A communication scheme for analog VLSI perceptive systems," *IEEE J. Solid-State Circuits*, vol. 30, pp. 660–669, June 1995.

[11] P. Venier, A. Mortara, X. Arreguit, and E. A. Vittoz, "An integrated cortical layer for orientation enhancement," *IEEE J. Solid-State Circuits*, vol. 32, pp. 177–186, Feb. 1997.

[12] S. Grossberg, E. Mingolla, and J. Williamson, "Synthetic aperture radar processing by a multiple scale neural system for boundary and surface representation," *Neural Networks*, vol. 8, nos. 7/8, pp. 1005–1028, 1995.

[13] B. E. Shi, "Gabor-type filtering in space and time with cellular neural networks," *IEEE Trans. Circuits Syst., I*, vol. 45, pp. 121–132, Feb. 1998.

[14] A. G. Andreou and K. A. Boahen, "Translinear circuits in subthreshold MOS," *Analog Integr. Circuits Signal Processing*, vol. 9, pp. 141–166, 1996.

[15] D. G. Nairn and C. A. T. Salama, "Current-mode algorithmic analog-to-digital converters," *IEEE J. Solid-State Circuits*, vol. 25, pp. 997–1004, Aug. 1990.

[16] A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Medeiro, and M. Delgado-Restituto, "High resolution CMOS current comparators: Design and applications to current-mode function generation," *Analog Integr. Circuits Signal Processing*, vol. 7, pp. 149–165, 1995.

[17] T. Serrano-Gotarredona, "VLSI implementation of a pseudo-gabor filter for edge extraction," M.S. thesis, The Johns Hopkins Univ., July 1997.

Teresa Serrano-Gotarredona for a photograph and biography, see p. 615 of the May 1999 issue of this Transactions.

Andreas G. Andreou, for a photograph and biography, see p. 616 of the May 1999 issue of this Transactions.

Bernabé Linares-Barranco for a photograph and biography, see p. 615 of the May 1999 issue of this Transactions.