A Programmable VLSI Filter Architecture for Application in Real-Time Vision Processing Systems

Teresa Serrano-Gotarredona[1], Andreas G. Andreou[2], and Bernabé Linares-Barranco[1]

[1]Instituto de Microelectrónica de Sevilla (IMSE), Centro Nacional de Microelectrónica (CNM),
Ed. CICA, Av. Reina Mercedes s/n 41012 Sevilla, SPAIN. Phone: 34-5-4239923,
Fax: 34-5-4231832, E-mail: terese@imse.cnm.es

[2]Dept. of Electrical and Computer Engineering, The Johns Hopkins University,
Baltimore, MD 21218, USA

Abstract

An architecture is proposed for the realization of real-time edge-extraction filtering operation in an Address-Event-Representation (AER) vision system. Furthermore, the approach is valid for any 2D filtering operation as long as the convolutional kernel *F(p,q)* is decomposable into an x-axis and a y-axis component, i.e. *F(p,q)=H(p)V(q)*, for some rotated coordinate system {p,q}. If it is possible to find a coordinate system {p,q}, rotated with respect to the absolute coordinate system a certain angle, for which the above decomposition is possible, then the proposed architecture is able to perform the filtering operation for any angle we would like the kernel to be rotated. This is achieved by taking advantage of the AER and manipulating the addresses in real time. The proposed architecture, however, requires one approximation: the product operation between the horizontal component *H(p)* and vertical component *V(q)* should be able to be approximated by a signed minimum operation without significant performance degradation. It is shown that for edge-extraction applications this filter does not produce performance degradation. The proposed architecture is intended to be used in a complete vision system known as the Boundary-Contour-System and Feature-Contour-System 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 electrical simulation and experimental results at the circuit level operation of some critical subcircuits.

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 or at different layers, and the way this is done is through physical connections called *synapses*.

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 Address Even Representation (AER) [1]-[3]. In AER each neuron codes its activity as a pulse stream signal with very low duty cycle, i.e. pulse width must be minimum but separation between pulses should be fairly large. Each neuron has a code or address, and every time it produces a pulse it will try to write its code on a common digital bus. A receiving system will continuously be reading this bus and send the pulse to those neurons who ought to be connected to the sending neuron. In this manner the activity of a large number of neurons can be time multiplexed on a common bus. This principle allows to structure hierarchically a very complex neural system. For example, a retina chip with AER output is continuously putting addresses on a bus representing the sensed images. Several chips, each with an AER receiver system, can be reading the same bus, doing some specialized processing and broadcasting the outputs of all their neurons using again AER on another external bus, and so on. Furthermore, extra processing can be added easily while the “addresses” go from one chip to the next. For instance, image rotation or translation can be performed in a straightforward manner by inserting an EEPROM for which the transformation operation has been programmed pixel by pixel (or address by address). In the architecture proposed in this paper we take advantage of this fact to simplify the processing chip.

As neuroscientists manage to unfold the internal structure and functions of the vision system, it becomes more feasible for mathematicians and computer scientists to propose and understand bio-inspired vision models and algorithms, and for engineers to build bio-inspired artificial vision systems. One powerful vision model proposed recently by Grossberg et al. [5] is the Boundary-Contour-System (BCS) and Feature-Contour-System (FCS) vision model. It consists of nine layers which after local illumination normalization and contrast enhancement of an input image, performs local edge extraction for different spatial orientations and scales, and then is able to identify consistent long range contours of the shapes in the input image through processing layers with feedforward and feedback connections. In this vision model one of the stages performs a 2D filtering operation for edge extraction, and other stages perform other 2D filtering operations. The processing architecture proposed in this paper is intended to be used in this vision model to perform a simplified version filter doing the

edge-extraction operation. The same processing architecture can be reprogrammed to perform some of the other 2D filters needed in the BCS-FCS vision model.

The present paper is structured as follows. In the next Section we will briefly describe the structure, functionality, and operations performed by the BCS-FCS vision model. In Section III we introduce modification of the edge-extraction kernel which substitutes a *product* operation by a *minimum* operation in the original kernel. Section IV describes briefly the essence of AER, and in Section V we introduce a VLSI architecture capable of implementing a 2D programmable filter. Section VI provides system level behavioral simulation results of this architecture programmed with a kernel to do an extraction of vertically oriented edges, and finally Section VII indicates the conclusions and future work.

II. The Boundary-Contour-System and Feature-Contour-System Vision Model

Fig. 1 shows a schematic representation of the structure of the BCS-FCS model [5]. The BCS consists of several identical subsystems (three in the case of Fig. 1) each of which is tuned for a different spatial scale. Each BCS spatial subsystem consists of 8 layers. Consecutive layers have been drawn in Fig. 1 as connected by thick shaded arrows. We may think of these arrows as the representation of a convolution (or filter) operation applied to the state of the previous layer and resulting in the state of the next layer. For instance, the 2D input image suffers three different filtering or convolutional operations, each of which is the starting point of a BCS subsystem. From here on, each BCS subsystem operates autonomously. From *Layer* 1 to *Layer* 3 there are only feedforward filtering operations, while *Layers* 4 to 8 are connected in a feedback loop configuration, which means the system will reach a steady state after a certain number of iterations (if the system is implemented sequentially on a computer) or after a certain time constant (if the system operates asynchronously and fully parallel, like in biological brains). The outputs of *Layers* 1 and 5 of the three BCS subsystems are fed to the FCS. Next we will briefly describe the processing performed on the different layers.

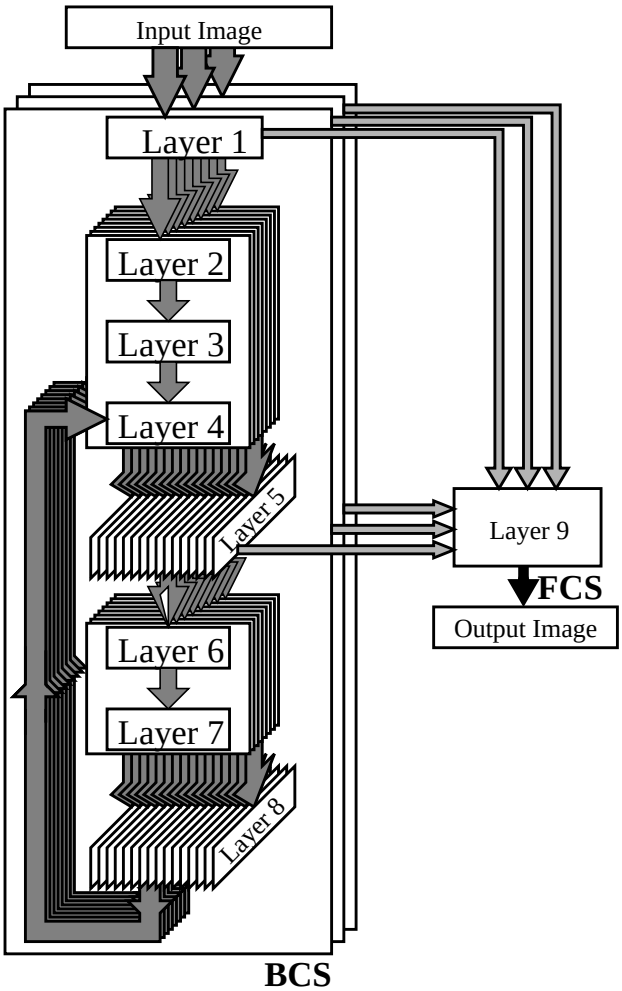


Fig. 1: Schematic Representation of Grossberg's et al. Boundary Contour System (BCS) and Feature Contour Systems (FCS) Vision Model

A.Stage 1: Center-ON OFF-Surround

Let us assume

$$I(p,q) \qquad \begin{cases} p = 1, \ldots N \\ q = 1, \ldots M \end{cases} \tag{1}$$

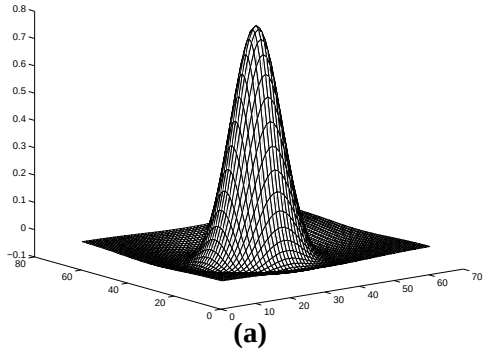
is an $N \times M$ input image provided by a vision sensing front end. This input image is applied to a 2D filter whose impulsive response or *kernel* of radial symmetry is shown in Fig. 2(a). We can see that pixels close to the center region of the kernel are going to contribute with positive weights to the convolution, while pixels further away will contribute negatively. The result of such a convolution is local illumination normalization and contrast enhancement. The mathematical expression for this kernel is

$$S_1(p,q) = A_1 e^{\frac{p^2+q^2}{\sigma_g}} - A_2 e^{\frac{p^2+q^2}{\alpha\sigma_g}} \tag{2}$$

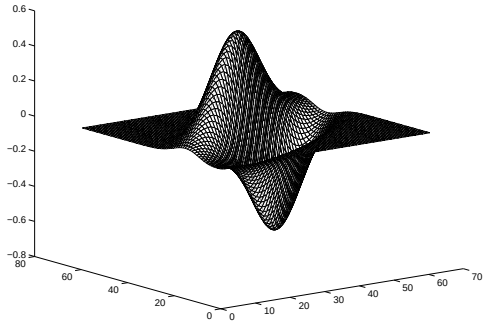
where $A_1 > A_2$, are positive parameters, $\alpha > 1$, and σ_g controls the spatial scale of the filtering. In the case of Fig. 1 there are three BCS subsystems, which means three Center-ON OFF-Surround filters are applied in parallel to the same input image, each with a specific σ_g ($g = 1, 2, 3$). From now on the processing in each BCS subsystem is independent.

B.Stage 2: Simple Cells

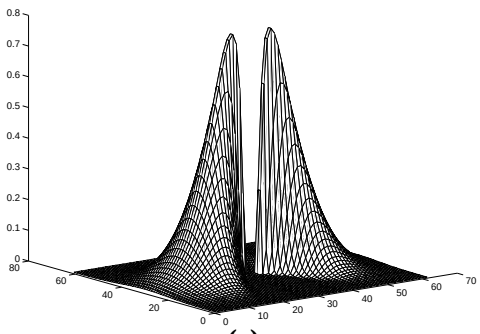
The second stage of the BCS system applies an orientation specific convolution for detecting edges oriented within a narrow angle range. This is performed by convolving the output of *Layer* 1 with the kernel shown in Fig. 2(b) for different orientations. This is why the output of *Layer* 1 in Fig. 1 suffers several convolutions in parallel, one for each orientation, resulting in as many "*Layer* 2" as orientations have been considered. The kernel of Fig. 2(b) is mathematically described by the difference between two displaced gaussians



(a)



(b)



(c)

Fig. 2: Convolutional Kernels used in the BCS system of Fig. 1: (a) Center-ON OFF Surround Kernel used by *Stages* 1, 4, and 7. (b)Edge-Extraction Kernel used by *Stage* 2. (c) Bipole Kernel used by *Stage* 6.

$$F_g(p_k, q_k) = \frac{1}{2\pi\sigma_{gh}\sigma_{gv}} e^{-\frac{1}{2}\left(\frac{p_k}{\sigma_{gh}}\right)^2} \left[e^{-\frac{1}{2}\left(\frac{q_k}{\sigma_{gv}}+\frac{1}{2}\right)^2} - e^{-\frac{1}{2}\left(\frac{q_k}{\sigma_{gv}}-\frac{1}{2}\right)^2} \right] \quad (3)$$

where the coordinate system (p_k, q_k) is rotated a certain angle with respect to the coordinate system of the input image (p, q),

$$\begin{aligned} p_k &= p\cos\frac{\pi k}{n_R} - q\sin\frac{\pi k}{n_R} \\ q_k &= p\sin\frac{\pi k}{n_R} + q\cos\frac{\pi k}{n_R} \end{aligned} \quad (4)$$

with n_R being the total number of orientations to be considered.

C.Stage 3: Complex Cells

After applying the filtering of *Stage* 2, a pixel in *Layer* 2 for orientation *k* will display a high positive value if the input image presents a positive change in contrast with respect to the *k-th* orientation axis. If the change in contrast is negative, the output of this pixel would be a high negative value. In order to detect whether or not there is an edge at that orientation around the given pixel there is no need to distinguish between positive and negative values. Therefore, the purpose of this processing stage is simply to rectify the output of the previous one.

D.Stage 4: Hypercomplex Cells, Competition across Space

At *Layer* 3 for orientation *k*, pixels will present a positive value if around that pixel there is an edge at that orientation. The higher the pixel value, the clearer the edge was. At this stage, and independently for each orientation, a 2D *Center-ON OFF-Surround* filter is applied to contrast enhance the previous image. This is equivalent to performing a spatial competition among pixels, favoring those with higher values.

E.Stage 5: Hypercomplex Cells, Competition across Orientations

At this stage, all *Layer* 4 pixels of the same spatial position but for all possible orientations *k*, are going to compete among them to contrast enhance those orientations with higher pixel values. This is done by applying a 1D *Center-ON OFF-Surround* filter to pixels of *Layer* 4 of the same spatial position but for all *k* orientation values.

F.Stage 6: Bipole Cells, Long-Range Cooperation

The operation of this stage is the most complicated. It tries to identify long term "*Contours*", which can be defined as *edges* that remain consistent over larger space ranges. This is achieved by performing for each orientation *k* the following sum of convolutions,

$$A^g_{pqk} = \sum_{r=1}^{n_R} C(p, q, k, r) \quad (5)$$

where A^g_{pqk} is the resulting state of pixel (p, q) of *Layer* 6 for spatial scale *g* and orientation *k*, subscript *r* denotes orientation, and each convolution is given by

$$C(p, q, k, r) = [y_r(p, q) - y_{\hat{r}}(p, q)] \otimes Z_r(p_k, q_k) \quad (6)$$

where $y_r(p, q)$ denotes the state of pixel (p, q) of *Layer* 5 for orientation *r*, $\hat{r}$ denotes orientation perpendicular to *r*, and the kernel is defined by

$$\begin{aligned} Z_r(p_k, q_k) &= \mathrm{sgn}(p_k) e^{-\beta(p_k^2+q_k^2)} e^{-\mu(q_k/p_k^2)^2} \times \\ &\times \cos\left\{ \frac{(k-r)\pi}{n_R} - \mathrm{sgn}(p_k)\,\mathrm{atan2}\left(\frac{2q_k}{p_k}, p_k\right) \right\} \end{aligned} \quad (7)$$

with β, μ, and γ being positive parameters. Fig. 2(c) depicts this kernel for the case $k = r = 0$.

G.Stage 7: Hypercomplex Cells, Competition across Space

This stage performs the same operation than *Stage* 4.

H.Stage 8: Hypercomplex Cells, Competition across Orientations

This stage performs the same operation than *Stage* 5. The output of *Stage* 8, *Layer* 8, is combined with the output of *Stage* 3, *Layer* 3, to form the input image for *Stage* 4. This way a feedback loop is formed, which once settled will yield the proper output of each BCS subsystem.

I.Stage 9: Feature Contour System (FCS)

The information about consistent long range contours can be taken from *Layer* 5 (once the feedback loop has settled), for all computed orientations. The FCS takes the original (local illumination normalized and contrast enhanced) image present in *Layer* 1 and performs a selective diffusion operation between pixels, using the contour information present at *Layer* 5: the contours at *Layer* 5 act as barriers to the diffusion operation. The result of all this processing is a clean noise-free image with clear and consistent long range contours.

III. An Edge-Extraction Filter

In the rest of this paper we will concentrate on *Stages* 2 and 3, the filter for edge-extraction and subsequent rectification. We will first introduce a simplification of the kernel of eq. (3) which will allow us to propose a very compact and efficient hardware that takes advantage of the AER as well.

The kernel of eq. (3) is decomposable into two factors, each of which depends only on either the x-coordinate p_k or the y-coordinate q_k,

$$F_g(p_k, q_k) = \frac{1}{2\pi} H_g(p_k) V_g(q_k), \tag{8}$$

with

$$H_g(p_k) = \frac{1}{\sigma_{gh}} e^{-\frac{1}{2}\left(\frac{p_k}{\sigma_{gk}}\right)^2}$$

$$V_g(q_k) = \frac{1}{\sigma_{gv}}\left[e^{-\frac{1}{2}\left(\frac{q_k}{\sigma_{gv}} + \frac{1}{2}\right)^2} - e^{-\frac{1}{2}\left(\frac{q_k}{\sigma_{gv}} - \frac{1}{2}\right)^2} \right] \tag{9}$$

The simplification proposed here consists in substituting the product operation between $H_g(.)$ and $V_g(.)$ by the signed minimum,

$$F_g'(p_k, q_k) = \frac{1}{2\pi} \mathrm{sgn}[H_g(p_k)] \mathrm{sgn}[V_g(q_k)] \times \\ \times min\{|H_g(p_k)|, |V_g(q_k)|\} \tag{10}$$

Fig. 3(b) shows the result of applying the filtering of eq. (8) to the input image of Fig. 3(a), while Fig. 3(c) results when using the kernel of eq. (10). As can be seen there is no appreciable difference in the resulting images.

To evaluate quantitatively the effect of the proposed approximation we can use the Normalized Square Error defined as,

$$NSE = \frac{\iint \|F(x,y) - F_m(x,y)\|^2 dxdy}{\iint \|F(x,y)\|^2 dxdy}. \tag{11}$$

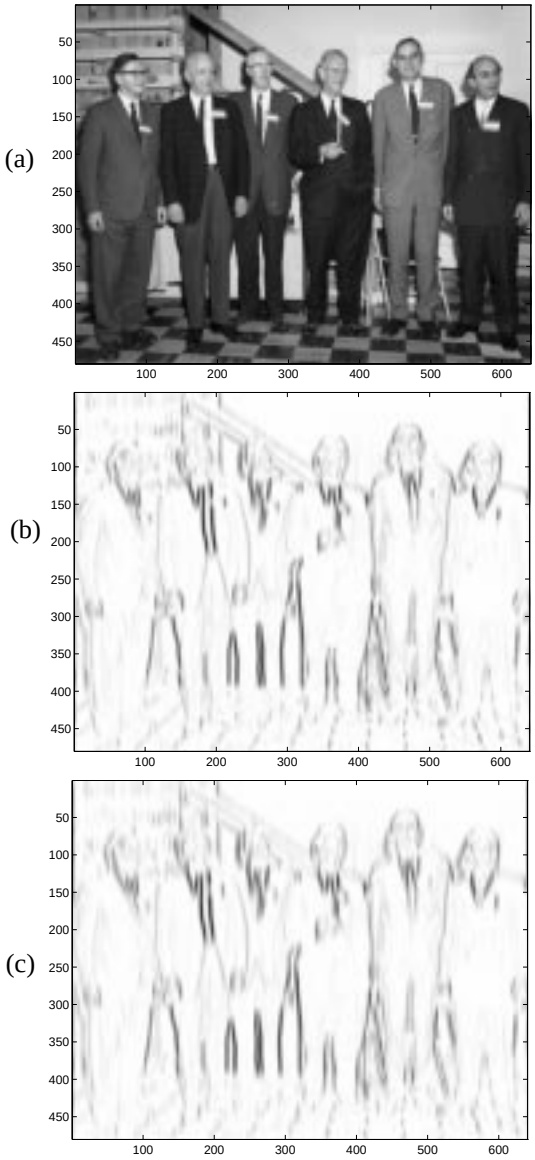


Fig. 3: Behavioral Simulation Results when performing 2D Filtering for vertical edge-extraction. (a) Input Image, (b) using the edge-extraction filter kernel of eq. (8), (c) using the modified filter kernel of eq. (10)

This quantity helps us to evaluate the difference between the original kernel $F(x,y)$ and the modified kernel $F_m(x,y)$ obtained when the product operation is substituted by the signed minimum. Table 1 gives the computed NSE for several kernels. All kernels in Table 1 are decomposable in the product of two functions that depend separately on the x and y components.

IV. Using Address Event Representation (AER)

Fig. 4 shows a schematic figure outlining the essence behind the AER. Suppose we have an "emitter" chip containing a large number of neurons or cells $D1, D2, D3,$ ... whose activity changes in time with a "relatively slow" time constant. For example, if *Chip* 1 is a retina chip and each neuron's activity represents the illumination sensed by a pixel, the time constant with which this activity

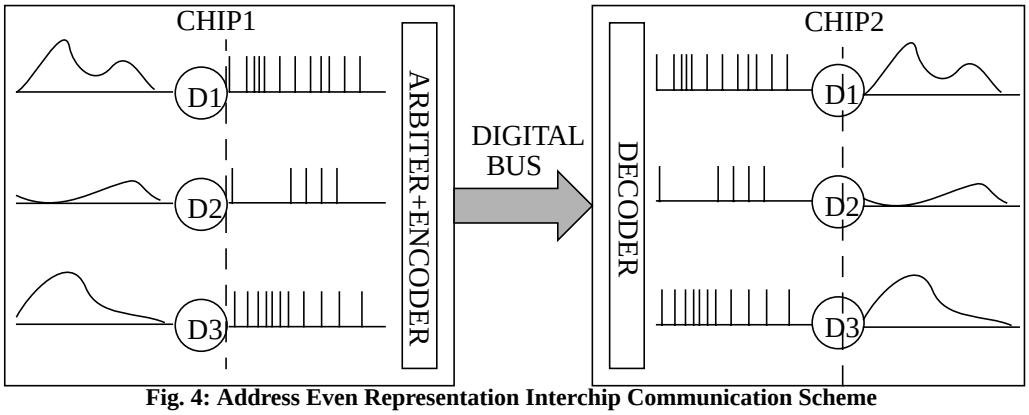


Fig. 4: Address Even Representation Interchip Communication Scheme

changes is, at the most, equivalent to *Frame-Rate* (i.e., 25-30 changes per second or a time constant of about 30-40*ms*).

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 more than one pulses are produced simultaneously by several neurons, a classical arbitration tree can be introduced [1]-[3], or one based in Winner-Take-All (WTA) row-wise competitions [6], or simply by making no neuron accessing the bus in case of a "collision" [7]. Whatever method is used the result will be the presence of a continuous 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) 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.

This is the simplest AER based communication scheme among chips. However, AER allows easily to 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 EEPROMs 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, and not in a time-multiplexed manner but in parallel. As we will see, the receiver chip will perform the following operations: for every address read on the bus it will send pulses to a bubble of neurons around that address. The width of those pulses is modulated according to some weights stored on chip. Time integration of those pulses for the complete array of neurons in the receiver chip implements a convolution operation. In the rest of the paper we will concentrate on describing the circuits able to implement such a convolutional or filtering chip.

Table 1

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 Design

Fig. 5 shows the basic operating principle of the proposed architecture. The address bus provides the coordinates (x_0, y_0) of the neuron (or pixel) around which the kernel of eq. (10) should be applied. Pulses will be applied to all rows with *y-coordinate* in the interval $[y_0 - L, y_0 + L]$, and all columns with *x-coordinate* in the interval $[x_0 - L, x_0 + L]$, where $2L + 1$ is the width considered for the kernel.

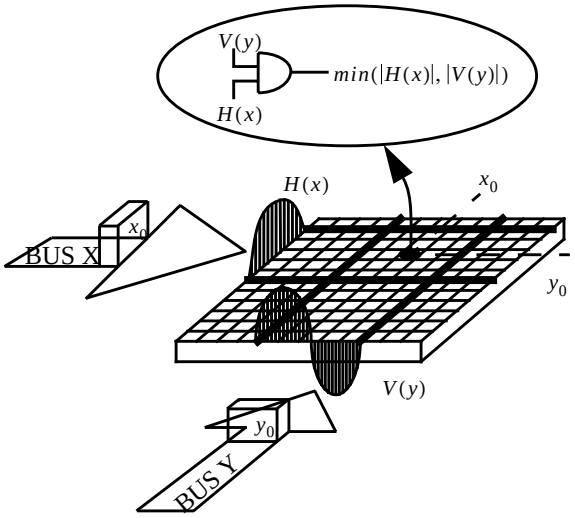


Fig. 5: Schematic Representation of the Basic Operation Principle behind the proposed Architecture

Pulses will be modulated in width according to function $|V(y)|$ (see eq. (8)) 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, called the "positive integrator", integrates the pulse of length $min(|H(x)|, |V(y)|)$ when $\mathrm{sgn}(H(x))\mathrm{sgn}(V(y)) > 0$; while the other, called the "negative integrator", integrates the pulse when $\mathrm{sgn}(H(x))\mathrm{sgn}(V(y)) < 0$. The values of $V(x)$ and $H(y)$ $(x, y = -L, \ldots 0 \ldots, L)$ are stored digitally on chip on a small RAM.

Fig. 6 shows the floorplan diagram of the system. It consists of two input decoders that decode the address of the arriving pulses, a C-element required for the AER communication protocol [1]-[3], an array of $N \times M$ integrator cells c_{ij}, two sets of programmable monostables $Mx_{-L}, ..., Mx_0, ..., Mx_L$ and $My_{-L}, ..., My_0, ..., My_L$ whose pulse widths are controlled by the bits stored in two RAMs, *RAM X* and *RAM Y* (which store the digital words $Rx_{-L}, ..., Rx_0, ..., Rx_L$ and $Ry_{-L}, ..., Ry_0, ..., Ry_L$, 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

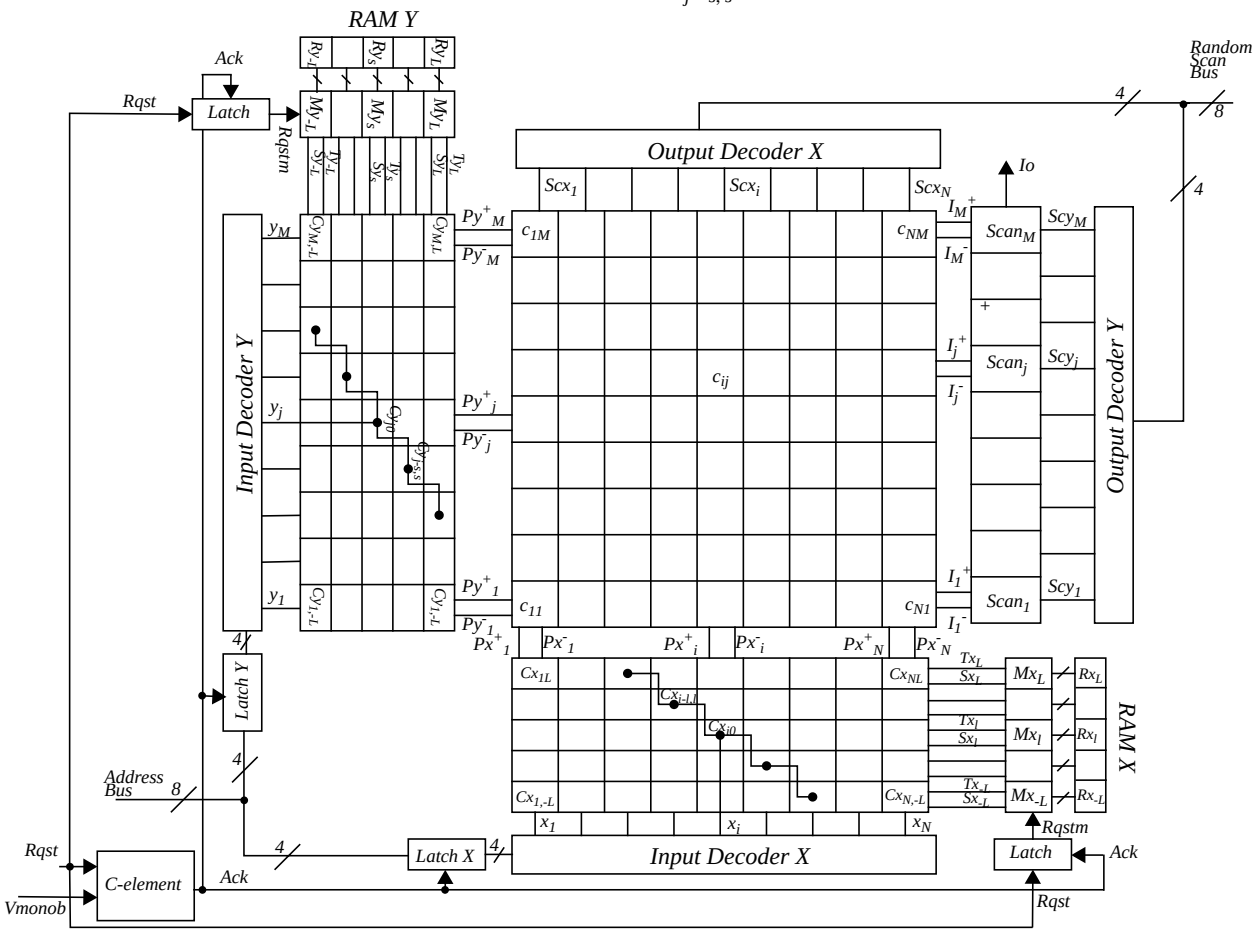


Fig. 6: Floorplan of Complete 2D Filtering System

cells to be scanned, and a column of scanning circuits $Scan_j$ to read out the integrators analog output current I_o. Note that in the present prototype of Fig. 6 the system does not generate an AER output. This can be solved by either adding the necessary circuitry to each pixel [1]-[3] which will decrease the cell density, or by adding a post-processing chip that scans sequentially all cells in the array of Fig. 6 and generates an AER output.

The operation of the system in Fig. 6 is as follows. In *RAM X* and *RAM Y* digital words of $n+1$ bits are stored (Rx_{-L}, ..., Rx_l, ..., Rx_L and Ry_{-L}, ..., Ry_s, ..., 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 pulses generated by the monostables are sent through lines Tx_l (or Ty_s) and are triggered whenever an external pulse arrives to the system (whenever signal *Rqst* pulses). 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. 6, $l = -L, \ldots 0 \ldots L$) connect the pulse in line Tx_l to line Px^+_{i-l} if the sign bit Sx_l is '1'. If the sign bit Sx_l is '0' line Tx_l is connected to the negative line Px^-_{i-l}. 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

$$I^+_{ij} = \sum_{\substack{p,q \\ \mathrm{sgn}(H(x_{pi}))\,\mathrm{sgn}(V(y_{qj}))>0}} I_w min(|H(x_{pi})|, |V(y_{qj}|))n_{pq} \quad (12)$$

where $x_{pi} = x_p - x_i$, $y_{qj} = y_q - y_j$, n_{pq} is the (lossy) integral over time of the number of pulses pixel (x_p, y_q) is receiving, 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 $(Px^+_i \cap Py^-_j) \cup (Px^-_i \cap Py^+_j)$. Hence, along time it computes the following sum

$$I^-_{ij} = \sum_{\substack{p,q \\ \mathrm{sgn}(H(x_{pi}))\,\mathrm{sgn}(V(y_{qj}))<0}} I_w min(|H(x_{pi})|, |V(y_{qj}|))n_{pq} \quad (13)$$

Consequently, the difference between the outputs of the positive and negative integrators is given by,

$$I_w \sum_{p,q} \mathrm{sgn}(H(x_{pi}))\,\mathrm{sgn}(V(y_{qj}))min(|H(x_{pi})|, |V(y_{qj})|)n_{pq}, \quad (14)$$

which is the filter operation we want to implement. In what follows we will describe the circuit components and operations of each block in Fig. 6.

A. Communication Protocol: The C-element

To perform a proper communication between two chips a communication protocol must be implemented [1]-[3]. In the AER scheme, the sender chip indicates when the address of a pulsing neuron is ready on the bus and the receiver chip must acknowledge that the pulse has been received and that it is ready to receive a new pulse.

Fig. 7 shows the timing diagram of a valid communication protocol for the two chips. The sender chip generates a request signal *Rqst* and the receiver generates an acknowledge signal *Ack*. When the sender has put the address on the bus it pulls the request signal *Rqst* to a high value. Once the receiver detects a high *Rqst* signal it latches the received address and pulls the acknowledge signal *Ack* high. The sender can put now *Rqst* low and begin to process the following pulse. The receiver must wait until all the monostables have sent their pulses to the corresponding neurons and the *Rqst* has gone low to put the *Ack* signal low. Once the *Ack* signal is low the sender can activate the *Rqst* signal to a high value again. Fig. 8 shows the schematic of the cell used in the receiver chip to generate the *Ack* signal. This cell is known as "C-element". This element receives two input signals: a request signal *Rqst* generated by the sender system, and signal *Vmonob* which is the wired-NOR of all the monostable output pulses $Tx_{-L}, \ldots, Tx_L, Ty_{-L}, \ldots, Ty_L$. The C-element generates an output acknowledge signal *Ack* which is sent back to the sender system. When no pulses are being received, *Rqst* is low. Signal *Vmonob* is high as no pulses are being

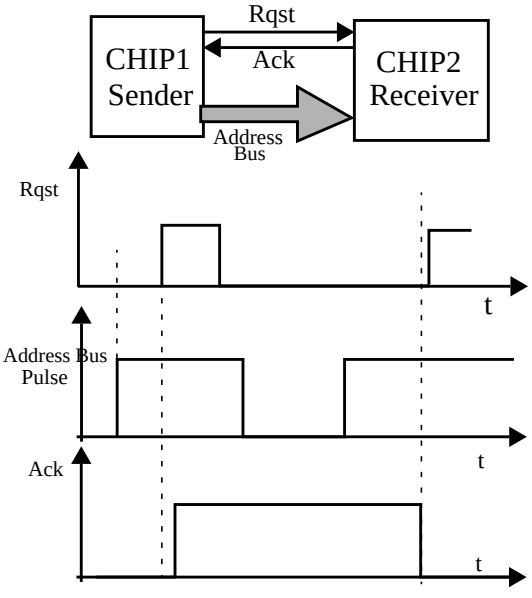


Fig. 7: Timing diagram of the address-event communication protocol

generated by the monostables. Consequently, the Ack signal is low. When a valid address pulse arrives the sender puts the $Rqst$ signal high. The rising edge of this $Rqst$ signal is used to trigger the monostables so that signal $Vmonob$ becomes low. Once $Rqst$ is high and $Vmonob$ has been set to low the C-element sets Ack to a high value, meaning that the $Rqst$ pulse has been received. The high value of Ack is used to latch the present bus address and signal $Rqstm$ that triggers the monostables. Latching the address assures that the corresponding neighborhood is kept selected until all monostables finish their pulses. By latching $Rqstm$ we assure that the monostable pulses do not end if signal $Rqst$ goes low before the monostable pulses have finished. The C-element waits until $Rqst$ goes low and all the monostable pulses finish ($Vmonob = 1$) to put the Ack signal low again. Once the acknowledge is low the sender is allowed to pull up $Rqst$ again and a new communication cycle can begin.

B.The Monostables

The schematic of a monostable with n controlling bits is shown in Fig. 9. Transistors $M1$ and $M2$ are equally sized, as well as transistors $M3$ and $M4$. Switches $S_1, \ldots, S_n$ are controlled by a digital *n*-bit word $b_1, \ldots, b_n$ that set the capacitance connected to node V_m. When no address is being received, Ack and $Rqst$ are both low and hence $Rqstm$ is also low. Transistor $M5$ is cut off so that node V_{out} is low. Node V_m is also set low through those transistors Mb_i with a high b_i value. If all b_i bits are low V_m will always be high (by $M8$) and no pulse will be generated. When an input pulse arrives signal $Rqst$ and hence $Rqstm$ become high. As soon as $Rqstm$ goes high node V_{out} goes high. Current I_T begins to flow through the switch formed by transistors $M6$ and $M7$ charging node V_m at the rate set by bits b_i. When node V_m reaches voltage value V_{thm}, the current through transistor $M2$ becomes higher than the current supplied by $M1$ so that the output node V_{out} flips from high to low. The length of the pulse at V_{out} is the time taken by current I_T to charge node V_m up to a voltage of V_{thm}. This time is given by

$$T = \frac{C_{mono}}{I_T} V_{thm} \qquad (15)$$

where C_{mono} is the total capacitance present at node V_m and is set by the bits $\{b_1, \ldots, b_n\}$ stored in the corresponding RAM word Rx_l or Ry_s. With this scheme, the length of the monostable pulses is linearly controlled between 0 and $(2^n - 1)C_u V_{thm}/I_T$, with n being the number of bits controlling each monostable pulse length, and C_u the unit capacitance in Fig. 9. Fig. 10 depicts the pulse widths obtained with Hspice versus the value of the digital control word, for a monostable controlled by $n = 5$ bits. For this simulation, values of $C_u = 0.2pF$, $I_T = 75\mu A$ and $V_{thm} = 2.5V$ were used.

C.The Selection Cell

Fig. 11 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. Each selection cell consists of two NAND gates controlling the gates of two PMOS transistors (MP^+ and MP^-) that behave like switches, and two NMOS pull down transistors (MN^+ and MN^- with a constant gate voltage V_{PD}). Each selection cell (for example, $Cx_{i-l,\,l}$ in Fig. 6) has two control signals (the decoder output x_i and the sign bit Sx_l from $RAM\ X$), 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 '1', 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 '0', line Tx_l is connected to the negative line Px^-_{i-l}. The same is valid for the Y coordinate selection cells.

D.The Core Cell

The schematic of cell c_{ij} of Fig. 6 is shown in Fig. 12. It consists of two diode-capacitor integrators [3]. The

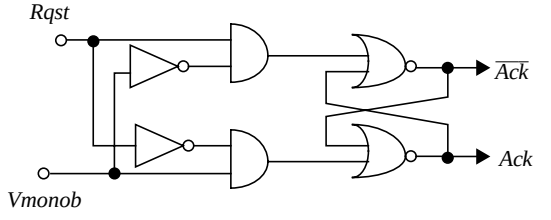


Fig. 8: Schematic of the C-element used for the arbitration of the address-event inputs

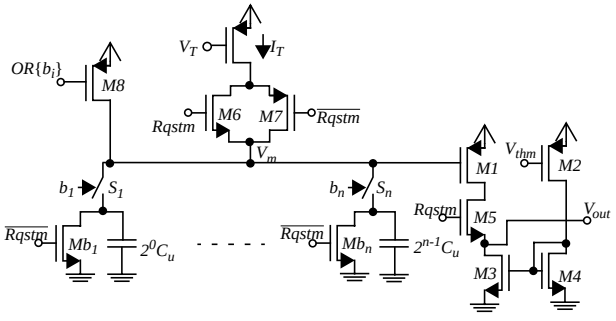


Fig. 9: Schematic of a Monostable Cell

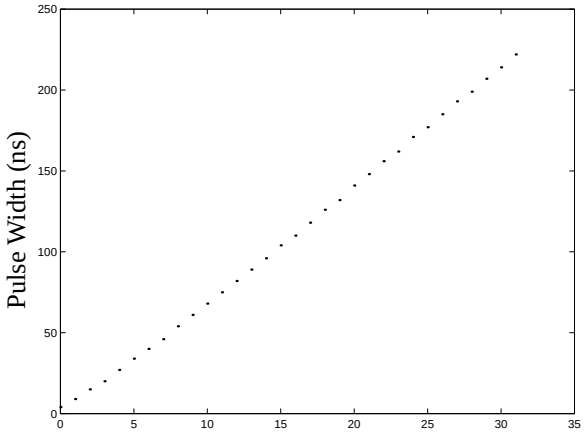


Fig. 10: Monostables pulse length expressed in nanoseconds versus value of controlling digital word obtained through Hspice simulation

positive integrator integrates the ANDED pulses that arrive in row and column lines with the same sign, that is, Px_i^+ and Py_j^+ (or Px_i^- and Py_j^-). The negative integrator integrates the ANDED pulses that arrive in row and column lines of opposite sign, that is, Px_i^+ and Py_j^- (or Px_i^- and Py_j^+). Each diode-capacitor integrator consists of two transistors M_1^+ and M_2^+ (M_1^- and M_2^-) operating in the subthreshold region, a capacitor C and a transistor M_w^+ (or M_w^-) acting as a current source of value I_w (controlled by bias voltage V_w) with its source pulsed by the output of the NOR gate. The input and output currents I_{in}^+ and I_{ij}^+ of the positive integrator are related to the voltage at node v_g^+ through the following differential equations (the treatment for the negative integrator would be the same for currents I_{in}^-, I_{ij}^- and voltage v_g^-) [3],

$$I_{in}^+ = -C\frac{dv_g^+}{dt} + I_{op}\exp\left(\frac{V_A - \kappa v_g^+}{v_t}\right) \tag{16}$$

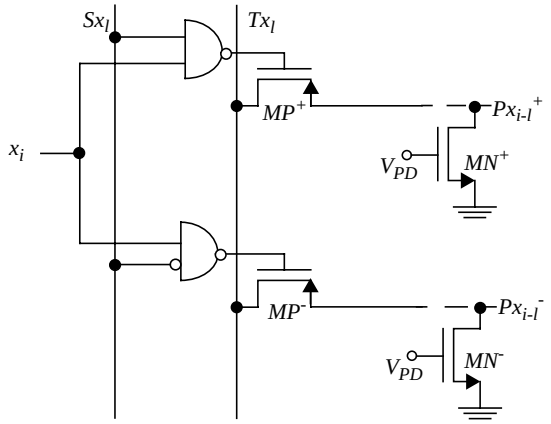


Fig. 11: Schematic of a neighborhood selection cell

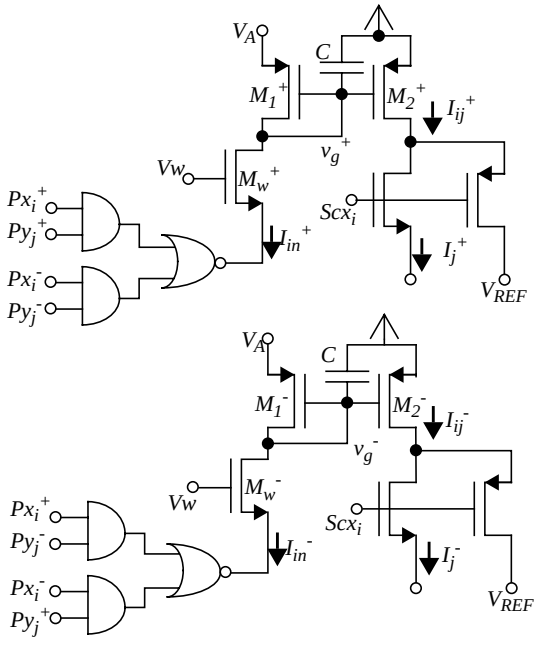


Fig. 12: Schematic of the Core Cell with the two diode-capacitor integrators

$$I_{ij}^+ = I_{op}\exp\left(\frac{V_{dd} - \kappa v_g^+}{v_t}\right) \tag{17}$$

where v_t is the thermal voltage and I_{op}, κ are model parameters of the MOS transistor operating in the subthreshold region. From eqs. (16) and (17) we can get an expression that relates the output and input currents of the diode

$$Q_T\frac{dI_{ij}^+}{dt} = I_{ij}^+\left(I_{in}^+ - \frac{1}{A}I_{ij}^+\right), \tag{18}$$

where

$$A = \exp\left(\frac{Vdd - V_A}{v_t}\right) \qquad Q_T = \frac{Cv_t}{\kappa} \tag{19}$$

Note that current mirror gain A is controlled by voltage V_A. During the time in which $Px^+{}_i$ and $Py^+{}_j$ (or $Px^-{}_i$ and $Py^-{}_j$) are simultaneously high, the source of transistor Mw^+ is low, and this transistor is acting as a current source sinking a constant current I_w from the integration node v_g^+. In this case $I_{in}^+ = I_w$ in eq. (18).

Suppose that a train of pulses of constant frequency $1/T$, pulse width T_h and interspike interval T_l (as depicted in Fig. 13) is applied simultaneously to lines $Px^+{}_i$ and $Py^+{}_j$ (or $Px^-{}_i$ and $Py^-{}_j$). Integrating equation (18) from t_1 to t_2 with $I_{in}^+ = I_w$, results in

$$\frac{1}{I_{ij}^+(t_1 + T_h)} = \frac{1}{AI_w} + \left(\frac{1}{I_{ij}^+(t_1)} - \frac{1}{AI_w}\right)\exp\left(-\frac{T_h}{\tau}\right), \tag{20}$$

where the integrator time constant is given by $\tau = Cv_t/\kappa I_w$. When the ANDED pulses are zero, the source of transistor Mw^+ becomes high and $I_{in}^+ = 0$. If the pulses go low at time t_2 and stay low for an interspike time T_l (see Fig. 13), the output current at time $t_2 + T_l$ just before a new pulse is applied, is given by

$$\frac{1}{I_{ij}^+(t_2 + T_l)} = \frac{1}{I_{ij}^+(t_2)} + \frac{T_l}{AQ_T}. \tag{21}$$

When pulses of width T_h are applied at a constant frequency $1/T$ as shown in Fig. 13, a steady state is reached in which the charge injected by the diode during the inactive periods equals the charge sank by the

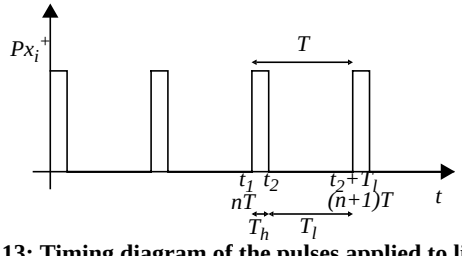


Fig. 13: Timing diagram of the pulses applied to lines Px_i^+, Px_i^-, Py_j^+ or Py_j^-

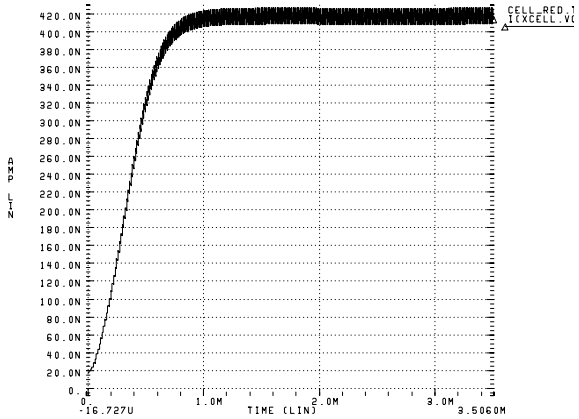


Fig. 14: Hspice Simulation of Integrator Cell

current source during the pulse. In this steady state, the two following equations must hold,

$$I_{ij}^{+}(t_1) = I_{ij}^{+}(t_2 + T_l) = I_{ijL}^{+}(\infty) \quad (22)$$

and

$$I_{ij}^{+}(t_2) = I_{ij}^{+}(t_1 + T_h) = I_{ijH}^{+}(\infty)\,. \quad (23)$$

Solving eqs. (20)-(23),

$$I_{ijL}^{+}(\infty) \cong AI_w\frac{T_h}{T_l} \cong AI_w\frac{T_h}{T}\,, \quad (24)$$

and the steady state ripple will be

$$\frac{\Delta I_{ij}^{+}(\infty)}{I_{ij}^{+}(\infty)} = \frac{I_{ijH}^{+}(\infty) - I_{ijL}^{+}(\infty)}{I_{ijL}^{+}(\infty)} \cong \frac{T_h}{\tau}\,, \quad (25)$$

where the assumption $\tau, T, T_l \gg T_h$ has been made.. According to equation (24), each integrator outputs a current which is proportional to the frequency $1/T$ and width T_h of the input pulses. Supposing 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 currents 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 non-rectified filter output.

Fig. 14 shows an Hspice transient simulation for one of the integrator cells in Fig. 12. Transistor sizes are $W = 12\mu m$ and $L = 12\mu m$, integrating capacitor is $C = 0.1pF$, pulse amplitude is $I_w = 13.5nA$, pulse width is $T_h = 100ns$, frequency of pulse stream is $1/T = 80KHz$, and voltage V_A was set to $4.67V$ (which yields a current gain from transistor $M_1^{+/-}$ to $M_2^{+/-}$ of around 2000).

To verify the operation of the diode-capacitor integrator a small prototype was integrated in a CMOS $2.5\mu m$ double-poly single-metal technology. The sizes of transistors M_1 and M_2 of the integrator were set to $W/L = 10\mu m/10\mu m$ and a the total capacitance at node

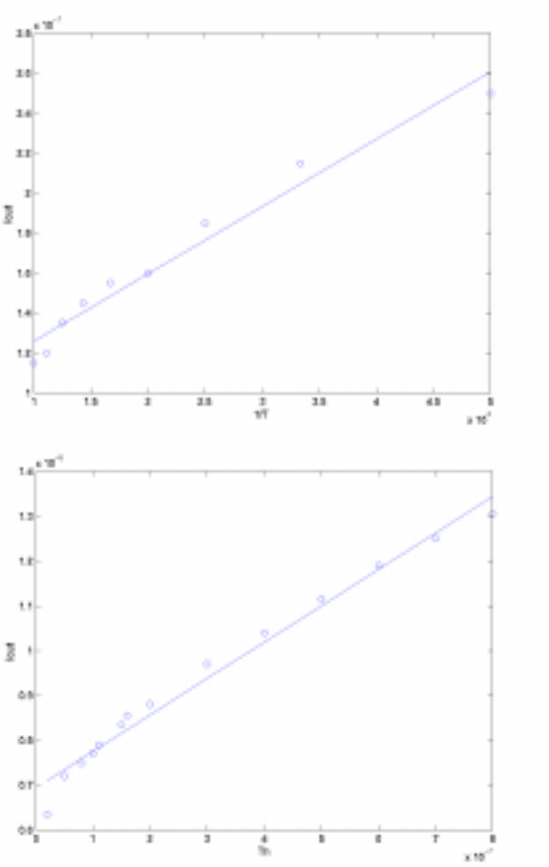


Fig. 15: Experimentally measured results of a diode-capacitor integrator, (a) Steady-State Current vs. frequency of Input Pulse Stream, (b) Steady-State Current vs. T_h

v_g was approximately $0.5pF$. To verify the linear dependence of the output current in the steady state versus the frequency of the arriving pulses and the width of the pulses (see eq. (24)), measurements of the output current in the steady state were performed where the frequency of the input pulse stream $1/T$ and the width of the pulses T_h were swept separately. The results are shown in Fig. 15. Fig. 15(a) shows the measured steady-state current level as a function of frequency. During the measurement voltage V_A was set to $4.7V$, the current bias $I_w = 10nA$ and the width of the arriving pulses was $T_h = 500ns$. Fig. 15(b) shows the steady-state current level as a function of pulse width, while maintaining the frequency constant at $100KHz$, for $V_A = 4.6V$ and $I_w = 10nA$.

E.The Scan Out Cell

A random access scanning circuit can read the rectified output current of any cell selected by the *Random Scan Bus* of Fig. 6. The output decoder X (see Fig. 6) selects a column *i*. When a column is not selected, the output currents I_{ij}^{+} and I_{ij}^{-} of all c_{ij} cells in that column flow to a line of constant voltage V_{REF}. If column i is selected, currents I_{ij}^{+} and I_{ij}^{-} of all c_{ij} cells in these columns flow to lines I_j^{+} and I_j^{-}, respectively, of the scan out cell $Scan_j$ shown in Fig. 16. Each scan out cell $Scan_j$ receives two input currents I_j^{+}, I_j^{-} and provides an output current $I_o = \left|I_j^{+} - I_j^{-}\right|$.

At the input of each scan out cell, current I_j^{-} is mirrored through a PMOS current mirror and subtracted

from current I_j^+. The PMOS current mirror has an active input [8] clamped to a voltage V_{REF}. This maintains a constant voltage V_{REF} at output nodes I_{ij}^- 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 $OPAMP_1$ [9]. It consists of an opamp ($OPAMP_1$) fed back with transistors M_1 and M_2 in a diode configuration. This feedback loop maintains the comparator input node (and output I_{ij}^+ of all selected c_{ij} cells) clamped to voltage V_{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_{REF}, thus transistor M_4 mirrors the current passing through M_2,

$$[I_j^- - I_j^+]^+ = \begin{cases} I_j^- - I_j^+ & if \quad I_j^- - I_j^+ > 0 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

The precision of this current reflection depends on how tightly the source of M_2 is clamped to voltage V_{REF}. To achieve a good precision a high gain opamp is needed. If current $I_j^+ - I_j^-$ is positive transistor M_1 sources this current, which is mirrored by transistor $M3$ because its source is clamped to V_{REF} by the current comparator composed by transistors $M5, M6$ and $OPAMP2$. Therefore, the current through $M3$ and $M5$ is,

$$[I_j^+ - I_j^-]^+ = \begin{cases} I_j^+ - I_j^- & if \quad I_j^+ - I_j^- > 0 \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

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 $I_o = \left|I_j^+ - I_j^-\right|$. Since transistors $M1 - M7$ operate in weak inversion, increasing the source voltage of transistors $M4$ and $M7$ with respect to V_{REF} will make the current mirrors $M2, M4$ and $M5, M7$ to have a gain higher than one (actually the gain will be exponentially controlled by this voltage difference).

Fig. 17 shows an Hspice simulation of the DC characteristic of a scancell. In this simulation, current I_j^+ was set to $80nA$ and current I_j^- was swept from $0nA$ to $160nA$. Two traces are shown in Fig. 17. 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. Note that a current amplification of about 150 has been applied from the input to the output currents. This allows speeding up the current read out process.

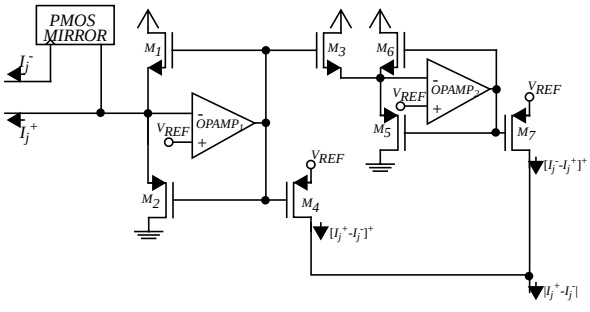


Fig. 16: Schematic of a cell to scan out the absolute value of the difference of two currents

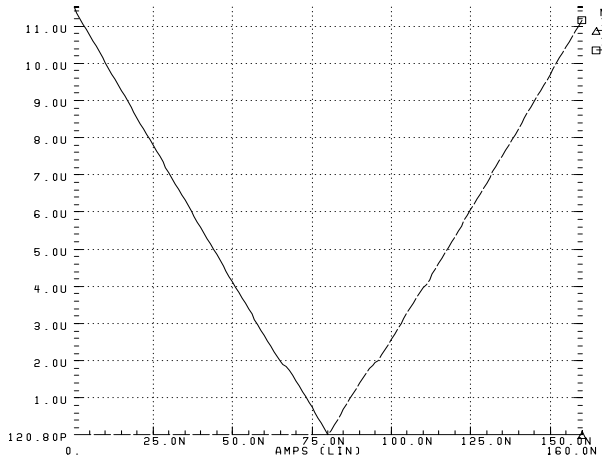


Fig. 17: Hspice DC input-output Characteristics of a Scancell

VI. System Level Operation Behavioral Simulations

So far electrical (Hspice) simulations and experimental measurements 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. 6 for a system of 128×128 cells. The input image fed to the system is shown in Fig. 18(a). 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 $(x_i(t_n), y_j(t_n), t_n) \quad n = 0, 1, 2, \ldots$ 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. 6. 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 eq. (18))

$$\begin{aligned} Q_T \frac{dI_{ij}^+}{dt} &= I_{ij}^+ \left(I_{in}^+ - \frac{1}{A} I_{ij}^+ \right) \\ Q_T \frac{dI_{ij}^-}{dt} &= I_{ij}^- \left(I_{in}^- - \frac{1}{A} I_{ij}^- \right) \end{aligned} \tag{28}$$

whose solution is of the form given by eq. (20) (to compute its charging during the presence of a pulse) or by eq. (21) (to compute its discharge during the absence of pulses). These equations 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 eq. (10) and the simulation results of Fig. 10. Depending on

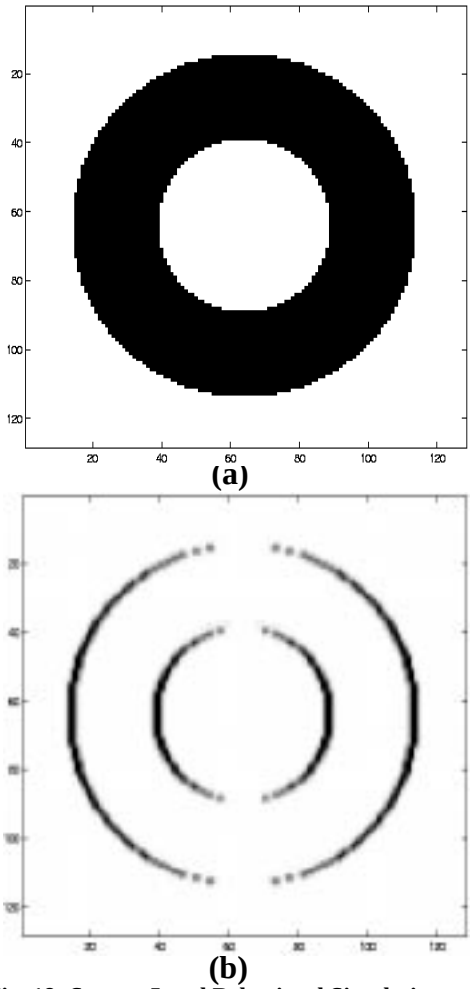


Fig. 18: System-Level Behavioral Simulations of a 128×128 Array. (a) Input Image, (b) Output Image of Pseudo-Gabor Filter extracting Vertical Edges

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 with eq. (21). For each cell c_{ij} its output is given by $\left|I_{ij}^{+} - I_{ij}^{-}\right|$. Using this method until all integrators have reached their steady state within 1% tolerance results in the system output depicted in Fig. 18(b). In this case, addresses were not pre-rotated, 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.

VII. Conclusions and Future Work

An architecture that implements a pseudo-Gabor filter for edge extraction has been presented. The architecture allows to implement any 2D 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 to rotate the 2D convolution kernel any angle.

A VLSI circuit implementation that realizes the proposed architecture is provided. Circuit simulation results and experimental measurements of critical components were given. System-level behavioral simulations of a 128×128 array have been included which validate the proposed approach. Cell size is $67.2\mu m \times 72.6\mu m$ if no AER output is available and $75\mu m \times 90.6\mu m$ if AER output is included, for a $1.2\mu m$ double-poly double-metal CMOS process. This would allow, for a $1cm^2$ die, to implement a 2D filter with approximately 128×128 pixels for no AER output, and 120×100 pixels if AER output is provided.

Future work includes the fabrication of a test prototype, its interface to a retina chip with AER output, and the implementation of more processing layers of the vision model system described in Section II. Note that the present architecture can be used to implement the processing of *Stages* 4, 5, 7, and 8 as well. For the implementation of *Stage* 6 the present architecture can be used if it is possible to substitute the kernel of eq. (7) by another one decomposable into horizontal and vertical components and if the product can be approximated by a signed minimum.

VIII. References

[1] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie, "Silicon Auditory Processors as Computer Peripherals," *IEEE Transactions on Neural Networks*, May, 1993.

[2] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*, Kluwer Academic Publishers, 1994.

[3] Kwabena Boahen, "Retinomorphic Vision Systems," *Microneuro'96: Fifth Int. Conf. on Neural Networks and Fuzzy Systems*, Lausanne, Switzerland, February 1996.

[4] Gordon M. Shepherd, *The Synaptic Organization of the Brain*, Oxford University Press, 3rd Edition, 1990.

[5] 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, No. 7/8, pp. 1005-1028, 1995.

[6] Z. Kalayjian, J. Waskiewicz, D. Yochelson, and A. G. Andreou, "Asynchronous Sampling of 2D Arrays using Winner-Takes-All Arbitration," *Proceedings of the 1996 IEEE Int. Symp. on Circuits and Systems* (ISCAS'96), Atlanta, vol. 3, pp. 393-396, 1996.

[7] A. Mortara, E. A. Vittoz, and P. Venier, "A Communication Scheme for Analog VLSI Perceptive Systems," *IEEE Journal of Solid-State Circuits*, vol. 30, No. 6, pp. 660-669, June 1995.

[8] D. G. Nairn and C. A. T. Salama, "Current-Mode Algorithmic Analog-to-Digital Converters," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 997-1004, August 1990.

[9] 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 Integrated Circuits and Signal Processing*, vol. 7, pp. 149-165, 1995.