A Scalable and Programmable Simplicial CNN Digital Pixel Processor Architecture

Pablo S. Mandolesi, *Member, IEEE*, Pedro Julián, *Member, IEEE*, and Andreas G. Andreou, *Member, IEEE*

***Abstract*—We propose a programmable architecture for a single instruction multiple data image processor that has its foundation on the mathematical framework of a simplicial cellular neural networks. We develop instruction primitives for basic image processing operations and show examples of processing binary and gray scale images. Fabricated in deep submicron CMOS technologies, the complexity of the digital circuits and wiring in each cell is commensurate with pixel level processing.**

***Index Terms*—Cellular neural networks (CNNs), CMOS imager sensor, digital pixel, image processing, pixel level processing, vision chips.**

I. Introduction

THE STILL and video camera community is seeing a renaissance as CMOS imagers are replacing charge coupled devices (CCD) sensors [1]. Fabricating sensor arrays in CMOS has the potential of incorporating processing beyond amplification on the chip and even at the pixel level much like it is done in human and animal eyes. A number of CMOS chips inspired by the function of biological vision systems have been reported in the literature [2].

The cellular nonlinear/neural networks (CNN) approach, introduced by Chua [3] abstracts biology at a high level. In CNN architectures, information processing is implemented through the evolution of a continuous-time nonlinear dynamical network with nearest neighborhood connectivity. The CNN universal machine (CNN-UM) is one of the earliest systems [4] that implemented CNN programmable functionality on a chip, including a programming language, compilers and simulators to aid the design of applications. Another example of CNN hardware implementation is the work by Nan *et al.* [5] and Dominguez-Castro *et al.* [6], which have merged a CNN-UM type processor and an imager front-end on a single chip that acquires and processes images in a parallel fashion. The operation mode of the chip is still conventional in the CNN sense, i.e., it relies on the steady state evolution of an analog dynamical network. However, unlike the original CNN-UM architecture, the system has a digital interface with an on chip 7-bit analog–digital (A/D) and digital–analog (D/A) converters improving the programmability and simplifying the interface to digital computers. The silicon retina architectures of Mead [7], Boahen and Andreou [8], Boahen, [9], Delbruck and Mead [10], and Culurciello and Etienne-Cummings [11] abstract biology at a lower level. Koch [12], Etienne-Cummings *et al.* [13], and Andreou and Boahen [14] have also implemented focal plane processing architectures that include processing beyond gain control and spatio-temporal filtering. Most of the above architectures have limited programmability. More recent work in address event representation (AER) based vision systems has yielded architectures and implementation of systems that are fully programmable in functionality [15], [16].

On the digital side, recent work has demonstrated the possibility of obtaining good quality A/D conversion inside each pixel. El Gamal *et al.* [17] points out that digital on-chip pixel processing promises many significant advantages while analog processing, suffers from poor scaling properties such as the reduction of the operating voltage and the increase of leakage current. The megapixel imager proposed in [18] has an A/D converter in every pixel, and it is able to acquire images at up to 10 000 frames per second. In other words, once every 100 μs, every pixel contains a digital value that encodes the light intensity in 8 bits. Such general-purpose imagers whose goal is the precise restoration of information are, on the other hand, power hungry and consume many orders of magnitude the power that biological systems and to a certain extend silicon models of biological function do. On the other hand, if power dissipation is not a concern, and there is a need for a general purpose computational sensor then digital pixel and processing imagers appear to be more attractive.

Digital approaches to computational vision processing units implement single instruction multiple data (SIMD) architectures that execute the same instruction on the entire array. The Programmable Versatile Large Size Artificial Retina (PVLSAR) 2.2, developed by Paillet *et al.* [19], contains a 50-transistor operational unit inside each pixel. Image processing on this chip is limited to the combination of very simple arithmetic and logic operations, most of them one bit long. The system is also capable of acquiring and processing 4-bit grayscale images and it can perform node operations

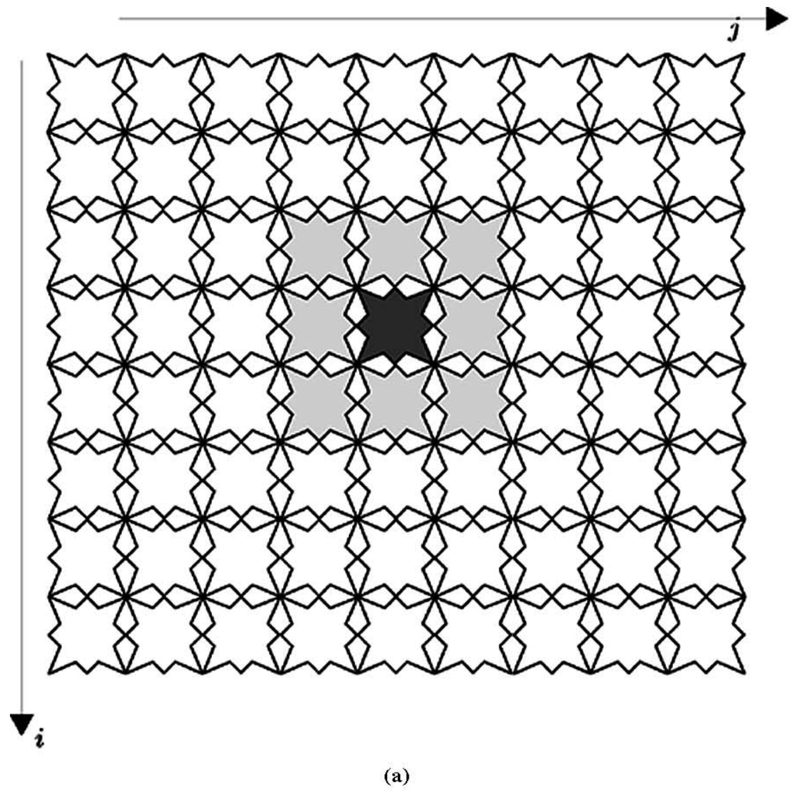
Manuscript received August 1, 2003; revised January 20, 2004. The work of P. S. Mandolesi and A. G. Andreou was supported in part by the National Science Foundation under Grant NSF-ECS-0010026 and Grant NSF-EIA-0130812. The work of P. Julián was supported in part by Estructuras de Procesamiento en Paralelo basadas en Redes Celulares No-Lineales of Fundación Antorchas, Argentina, under Project 14169/12. This paper was recommended by Guest Editor A. Zárandy.

P. S. Mandolesi is with the Department of Electrical and Computer Engineering, The Johns Hopkins Univiersity, Baltimore, MD 21218 USA, on leave from the Dipartimento de Ingenieria Eléctrica y Computadoras, Universidad Nacional del Sur, 8000 Bahía Blanca, Argentina (e-mail: pmandolesi@ieee.org).

P. Julián is with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA and also with Consejo Nacional de Investigaciones Cientificas y Técnicas (CONICET), Capital Federal CP 1033, Argentina, on leave from the Dipartimento de Ingenieria Eléctrica y Computadoras, Universidad Nacional del Sur, 8000 Bahía Blanca, Argentina (e-mail: pjulian@ieee.org).

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

Digital Object Identifier 10.1109/TCSI.2004.827626



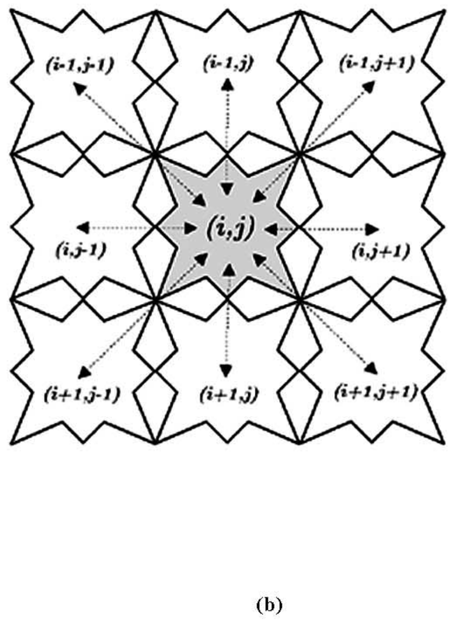


Fig. 1. (a) S-CNN cell structure in 2-D. (b) Array of cells in a sphere of influence.

with any combination of neighbors by employing an internal communication network based on circular shift registers.

In this paper, we propose a programmable architecture for a SIMD image processor that is based on the mathematical framework of simplicial CNNs (S-CNN). The S-CNN was originally introduced in [20] and contains in its core a simplicial piecewise-linear (PWL) function [21]. The S-CNN architecture is able to perform any arbitrary logic function: it can process gray or binary images and allows for complex image processing tasks to be programmed in an intuitive way.

The paper is organized as follows. In Section II, we introduce the basic architecture of a S-CNN, and the core processing functional blocks are presented in Section III. We develop instruction primitives for basic image processing functions and show examples of processing binary and gray scale images in Section IV. A brief discussion and conclusions follow in Section V.

II. S-CNN Basics

A CNN is a parallel computational structure defined on an array of interconnected cells. The array is homogeneous resulting in a displacement invariant lattice with local interconnects. The computation is based on the processing capabilities available at each cell; typically, a dynamical system with a local state, inputs and outputs.

The cell operates on the input and the local state of a particular set of neighbor cells (the neighborhood in general contains the cell itself) called sphere of influence (see Fig. 1). In an S-CNN, the differential/difference equation that governs the cell dynamics, utilizes a piecewise-linear function defined over a simplicial partition of the domain [20]. In this paper, we will restrict our attention to discrete-time S-CNN.

For arrays in two dimensions (2-D), each cell is described by the CNN equation ([3], [22])

$$\begin{cases} x_{i,j}(k+1) = F(\mathbf{y}_{S_{i,j}}(k), \mathbf{u}_{S_{i,j}}(k)), \\ y_{i,j}(k) = G(x_{i,j}(k)), \quad i = 1,\ldots,N, \qquad j = 1,\ldots,M \end{cases} \tag{1}$$

where $F : \Re^m \to \Re$ is a PWL function; $G : \Re \to [0,1]$ is the output function;[1] $\mathbf{y}_{S_{i,j}}(k) \in \Re^n$ is the vector of outputs corresponding to cells in the sphere of influence $S_{i,j}$; $\mathbf{u}_{S_{i,j}}(k) \in \Re^n$ is the vector of inputs corresponding to cells in the sphere of influence $S_{i,j}$; $x_{i,j}(k) \in \Re$ is the state of the (i,j) cell; N and M are the total number of rows and columns, respectively; n is the number of cells in the sphere of influence $S_{i,j}$, and $m = 2n$ is the dimension of the input/output (I/O) space.

The sphere of influence $S_{i,j}$ contains the cells that share input and state values with cell (i,j). In this paper, we will employ a nine-cell sphere of influence, i.e.,

$$S_{i,j} = \{(i-1,j-1),(i-1,j),\ldots,(i,j),\ldots,(i+1,j+1)\}$$

that results in nine-dimensional input and output vectors ($n = 9$). The resulting vector expressions are

$$\begin{aligned} y_{S_{i,j}} &= [y_{i-1,j-1}, y_{i-1,j}, \ldots, y_{i,j}, \ldots, y_{i+1,j+1}] \\ u_{S_{i,j}} &= [u_{i-1,j-1}, u_{i-1,j}, \ldots, u_{i,j}, \ldots, u_{i+1,j+1}] \end{aligned}$$

[1]In the standard CNN nomenclature, the output function is generally defined as $G(x_{i,j}) = (1/2)(|x_{i,j}+1| - |x_{i,j}-1|)$.

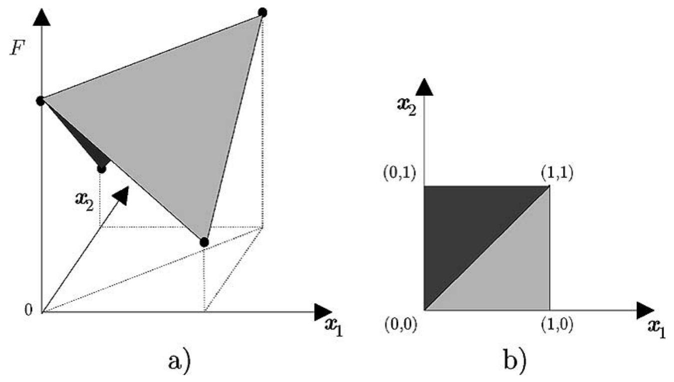


Fig. 2. (a) Simplicial PWL function. (b) Domain in $\Re^2$ with $\delta = 1$.

where $y_{k,l} \in [0,1]$, $u_{k,l} \in \Re$, $k \in \{i-1, i, i+1\}$ and $l \in \{j-1, j, j+1\}$.

Function F is PWL, and it is defined over a simplicially subdivided domain of the CNN I/O space, as can be seen from (1). A simplicial domain is a domain subdivided into simplices, where a simplex is an n–dimensional generalization of a triangle in $\Re^2$ or a pyramid in $\Re^3$ (Fig. 2). In every simplex, the descriptive function F is affine linear and can be represented by a hyperplane. A simplex is described by the convex combination of $m+1$ vertices. The vertices are the domain points defined by the partition intersections of zero dimension . If the partition H has a grid step δ then, the set of vertices is

$$\begin{aligned} V_D = \{\mathbf{v} \in \Re^m : v_i = p_j \times \delta; \\ i = 1, \ldots, m;\ p_j \in Z, 0 \le p_j \le p, p \times \delta = 1\}. \quad (2) \end{aligned}$$

Restricting the grid to the case $\delta = 1$, the vertices components can only take two possible values: zero or one; then, the set of simplices is

$$\begin{aligned} \widetilde{V}_D = \Big\{ \widetilde{V}_D^k = \left\{\mathbf{v}^1, \ldots, \mathbf{v}^i, \ldots, \mathbf{v}^{m+1}\right\} : k = 1, \ldots, m!, \\ v_j^1 = 0,\ j = 1, \ldots, m; \quad v_j^{m+1} = 1,\ j = 1, \ldots, m; \\ \left\|\mathbf{v}^{i+1} - \mathbf{v}_2^i\right\| = 1,\ i = 1, \ldots, m \Big\}. \quad (3) \end{aligned}$$

It was shown by Julian *et al.* [23] that the space of PWL functions defined over a simplicial partition is a Hilbert space, which in our case, has a dimension $q = 2^m$. We will use the basis proposed in [21], where the descriptive function is represented in the form of

$$F(w) = c^T \Lambda(w)$$

where $w \in \Re^m$ is a point in the I/O space, $c \in \Re^q$ is a coefficient vector, and $\Lambda : \Re^m \to \Re^q$ is a vector of basis functions, i.e., $\Lambda(\cdot) = [\alpha_1(\cdot), \ldots, \alpha_q(\cdot)]$. One of the advantage in representing the vectors in this space is that all elements $\alpha_i : \Re^m \to \Re^1$ have only one value different from zero over the set of vertices, i.e.,

$$\alpha_i(\mathbf{v}_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \quad \mathbf{v}_j \in V_D, \qquad i, j = 1, \ldots, q.$$

The information needed to represent the function is given by the values of the vertices (which coincide with the coefficients c_i) that can be stored in a table indexed by the binary number associated to the vertex coordinates.

A. S-CNN Computation

The S-CNN computes by solving the dynamical system (1). This necessitates that function F must be calculated in each cell for the system to evolve from time k to time $k+1$. We calculate the function F using the information stored in the indexed table and an algorithm to interpolate the final value. The procedures consists of three steps.

Step 1) Given an input vector w the simplex containing it is first identified. This simplex is described by the set of vertices $\widetilde{V}_D^k$.

Step 2) The vector consisting of the convex combination of the vertices $\widetilde{V}_D^k$ is then calculated. This corresponds to finding the coefficients μ_l of

$$w = \sum_{l=1}^{m+1} \mu_l v_l^k, \qquad v_l^k \in \widetilde{V}_D^k$$

where $\mu_l \in \Re, l = 1, \ldots, m+1$ and the $\sum_{l=1}^{m+1} \mu_l = 1$.

Step 3) Given the vertices in $\widetilde{V}_D^k$ and the convex combination weights μ_l, the values of the function F at the vertices are retrieved, i.e., $c_l,\ l = 1, \ldots, m+1$ and weighted to obtain the function value

$$F(w) = \sum_{l=1}^{m+1} \mu_l c_l.$$

An elegant way to implement the algorithm outlined above was proposed by Chien and Kuh in [24]. A modified version amenable to mixed analog/digital signal processing was described in [25].

In Section III, we propose an alternative architecture to map the algorithm into hardware, shown in Fig. 3.

III. Digital Architecture

As in previous implementations, the algorithm performs the calculations in a given time step. The digital inputs are supplied to a group of digital comparators and compared against a digital ramp. The output of each comparator is encoded in time by a 1-bit digital signal. The individual components $\widetilde{w}_i$, $i = 1, \ldots, q$ are grouped in a vector W that identifies the different vertices of the simplex containing w as the ramp goes through one cycle. The weights of the convex combination correspond to the amount of time each vertex appears at the comparator outputs. Note that the group of comparators calculate the convex decomposition described in the first two steps of the algorithm. The memory and the counter perform the third step. While the comparators produce the vertex number (used to address the memory) the corresponding value (c_l) is added in the counter. The count is updated in every ramp step so that at the end of the ramp cycle the value of the function is stored in the counter. This functionality is replicated in each cell, and hence the computation is done in parallel.

We assume that the processor will be implemented as an array of cells, so that all computations can be done in parallel.

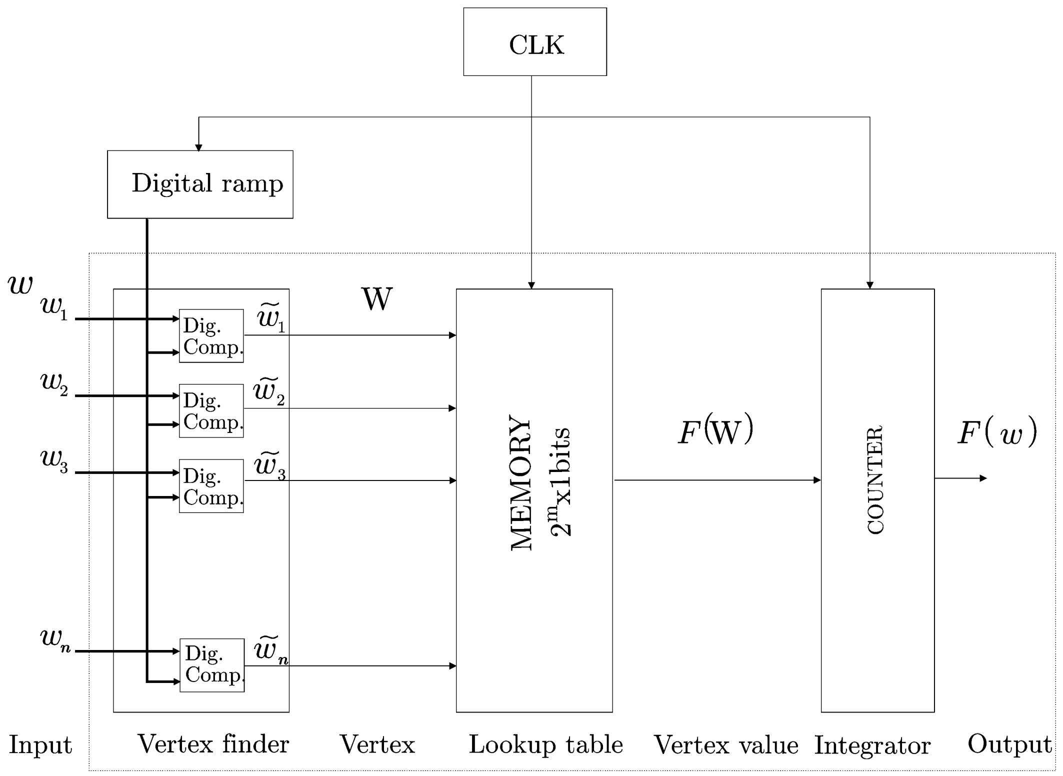


Fig. 3. Calculation of the function F using digital hardware.

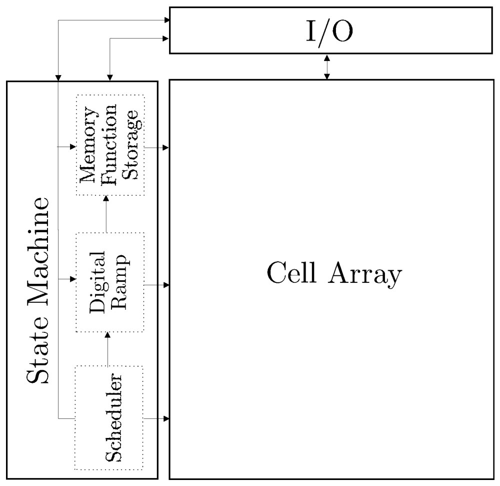


Fig. 4. S-CNN processor architecture. The I/O Interface contains the circuits to interact with the periphery of the chip. The I/O interface allows the readout of the whole array at every frame. It also incorporates the read-write circuits for accessing the memory where the functions and state machine code are stored.

Taking advantage of scaling in deep submicron CMOS technologies (90 nm and below), the S-CNN processor can even be included in the pixel circuit. The resulting structure could be a sensor array with local digital processing, capable of real time still image or video processing. The S-CNN processor consists basically of three parts: an array of basic cells, and the state machine unit as well as the I/O interface unit in the periphery. Fig. 4 shows a block diagram of the S-CNN processor architecture.

We now proceed to discuss the details of the basic cell in the array aimed at an implementation where wiring complexity is traded off to local storage by removing individual program storage in each cell (see Fig. 3) and by employing a single memory for the entire array. This architectural decision has two distinct advantages: First, the circuit complexity and the size of every cell is reduced and second, it offers more flexibility for storing and changing on the fly the S-CNN programs. To distribute the instruction, the memory content is wired through a parallel 9 bit global bus to all cells. This bus is used to broadcast the instruction (memory content) once at each ramp step, therefore adding an internal cycle, called evaluation cycle, at each step of the digital ramp.

Each cell in the array has the necessary circuits to:

1) acquire and convert the image into digital data (**PhotoADC**);
2) encode the data and communicate to the neighbors (**LocalComm**);
3) compute the state (**LocalComp**);
4) communicate with the periphery to output results (**CellInOut**).

These functions are done by the blocks depicted in (Fig. 5) as follows.

1) **PhotoADC:** The basis for this functional block is the digital pixel design of [18]. The circuits in this block include the photosensitive element and associated A/D conversion. Photo generated current is integrated on a capacitor to give an output voltage. The voltage is sampled, held, and compared to an external analog ramp that runs synchronously with a digital ramp. When the ramp voltage is greater than the photodiode voltage, the comparator changes its output and the value of the digital ramp is

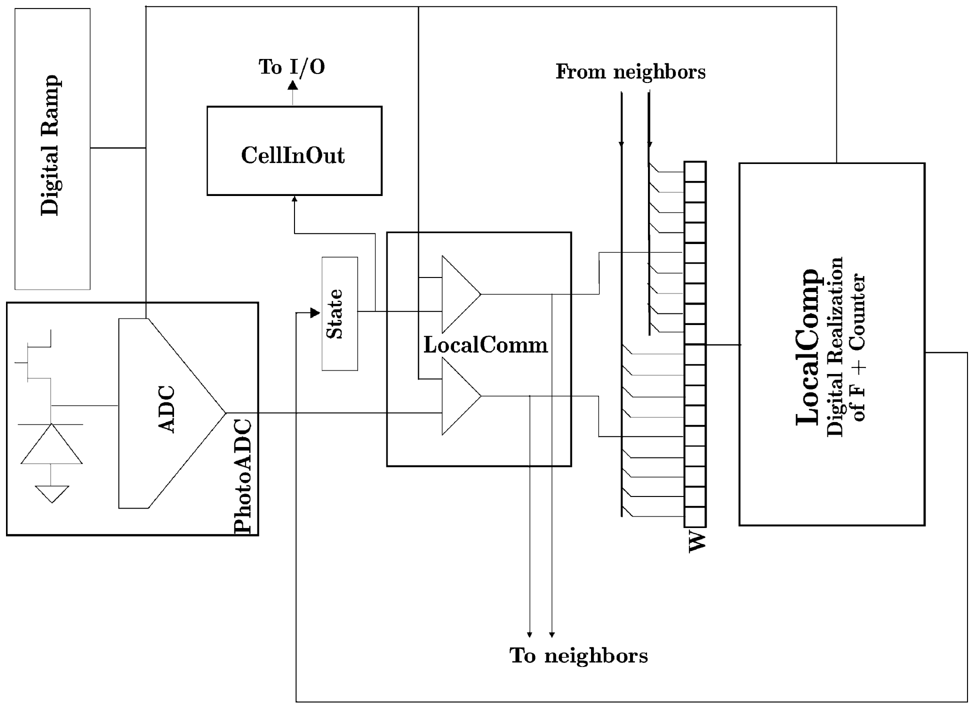


Fig. 5. S-CNN processor block diagram.

latched. In this way, the A/D conversion is performed and the input digital value is stored in a register file.

2) **LocalComm**: The encoder block contains the circuitry required for the time encoding of the input, the time encoding of the state values and their communication with the neighboring cells. Every signal is encoded with one bit, in such a way that it is zero when the input (state, respectively) is greater than the cycle ramp and one otherwise. The encoders are two digital comparators. They compare the values of both, the digital input u and the digital state x, with the digital ramp. As a result, two signals are obtained that are shared with all neighbors in the sphere of influence. Every cell collects nine pairs of these encoded signals, one per neighbor. The wiring for the digital ramp is shared with the wiring for the digital ramp required in the PhotoADC block. This has important ramifications in the power dissipation for the system (see discussion later in this section).
3) **LocalComp**: The local processing block groups the vectors corresponding to the inputs and states of the cells in the sphere of influence in a digital time varying word W. At each step of the cycle ramp the value of function F at the vertex indicated by W is integrated. At the end of the ramp cycle, the integration counter holds the new state value x. These tasks are done with two digital comparators. The first one compares W with the evaluation cycle ramp and detects when the broadcasted function corresponds to the vertex in W (Fig. 6). This comparator triggers a latch signal to a flip-flop that holds the value of the function. The value stored in the flip-flop triggers an increment of a counter at the end of the evaluation cycle.
4) **CellInOut:** The input/output block include the circuits necessary to enable the reading or writing of the cell state from the periphery of the chip.

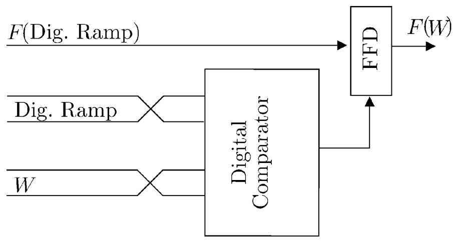


Fig. 6. Block diagram of the cell function evaluation.

The state machine contains the circuits responsible of scheduling the time and the execution order of the different events: the integration time, the conversion time, the I/O time and the ramp cycle with its internal evaluation cycle. It is also responsible of broadcasting the digital ramp for the analog conversion, the ramp cycle and the evaluation cycle. It can execute multiple S-CNN cycles in one frame and select the function from different memories. This can be defined by the user through programming the state-machine code. As every instruction operates over the whole cell array, the architecture can be also classified as a SIMD processor.

IV. S-CNN Programming

In this section, we present the basic programming primitives for the proposed S-CNN architecture. We will show that tasks can be programmed in a very intuitive way by writing down a truth table. This is an important advantage over analog implementation of CNN architectures, where, in most cases, the design of the differential equation for a given task is the result of a trial and error process or an iterative algorithm.

In the architecture presented in this paper to program an application, we only need to identify the relation between a cell state at $t = k + 1$ and the actual inputs and states of those cells

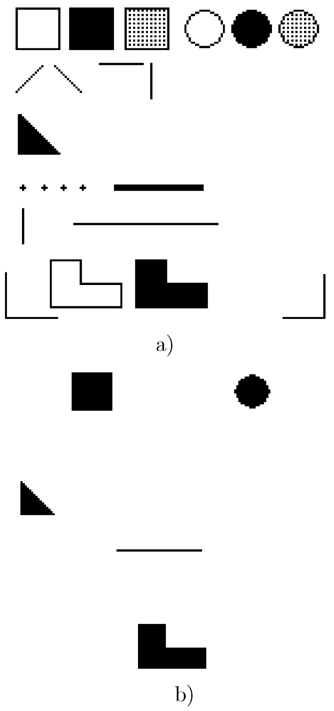


Fig. 7. (a) Input image. (b) Output result from the S-CNN erosion program.

in its sphere of influence at $t = k$. The domain of this functional mapping is, in general, the input output space of the S-CNN. The function F is completely defined by the values of the function at the set of points called vertices. To identify the function, we enumerate all the possible vertices or all possible combinations of u and x that belong to a vertex and for each one we assign the desired output state of the cell. Example 1 shows a simple case of a program that can be completed in one step. More complex programs can be written using a combination of one-step programs and/or loading special initial states or border conditions. However, in all cases, the programming methodology is the same and the programs are designed based on the primitives for the one-step program necessary to arrive to the desired steady state after a number of steps. Example 2 shows an example of a multi-instruction program.

Example 1: Find the function F for the S-CNN program *erosion* (Fig. 7). We first note that this necessitates feedforward only processing, that only depends on the input image. Therefore, the number of vertices with different values is reduced to 512 (2^9). If the cell input is zero, then, the state will also be zero, i.e., $x_5^{k+1} = 0$ if $u_5^k = 0$. For the remaining 256 possible input patterns, i.e., with $u_5^k = 1$, the new state will be one only when all the neighbors are one, i.e., $x_5^{k+1} = 1$ if $u_i^k = 1$, for all $i \in \{1,2,3,4,6,7,8,9\}$. Combining the two cases, a closed-form expression for the function F is obtained

$$x_5^{k+1} = u_5^k \cdot \left(u_1^k \cdot u_2^k \cdot u_3^k \cdot u_4^k \cdot u_6^k \cdot u_7^k \cdot u_8^k \cdot u_9^k\right).$$

Example 2: Find function F for the S-CNN program *hole-filling* (Fig. 8).[2] In this application, we use the erosion S-CNN

[2]Note that as circles plotted using single point lines include diagonal line segments as boundaries, and thus do not qualify as closed figures when a 3 × 3 neighborhood is considered.

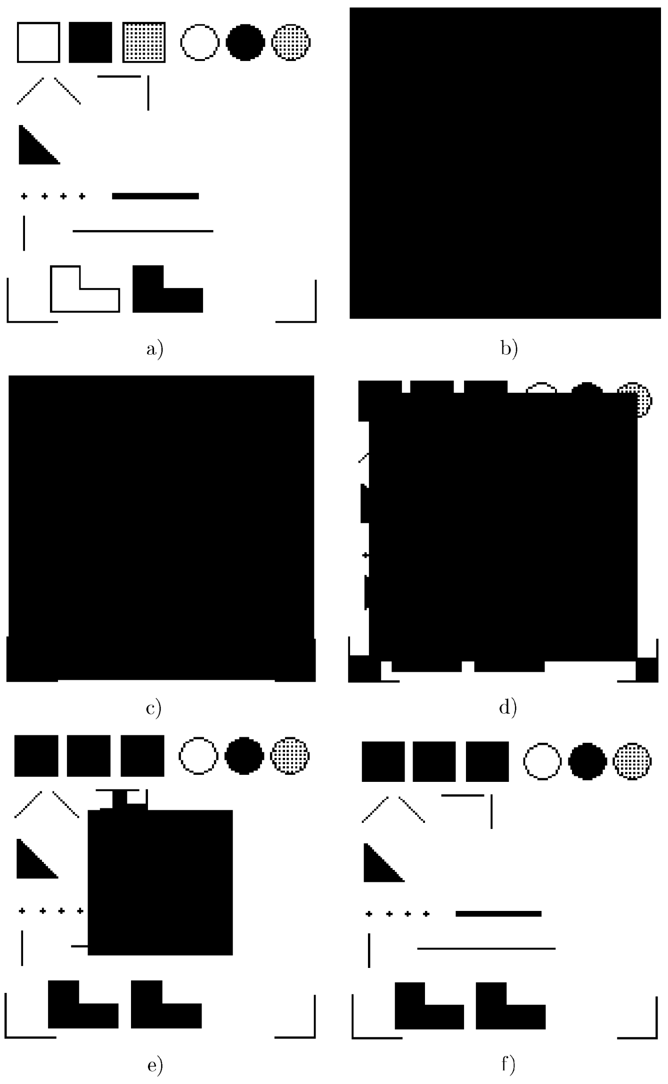
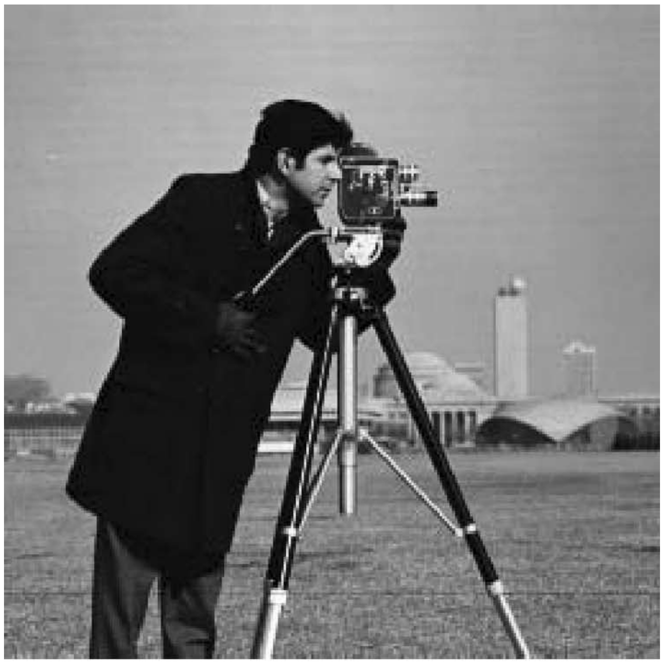


Fig. 8. (a) Input image. (b) Initial state. (c) State at $k = 1$. (d) State at $k = 10$; .... f.

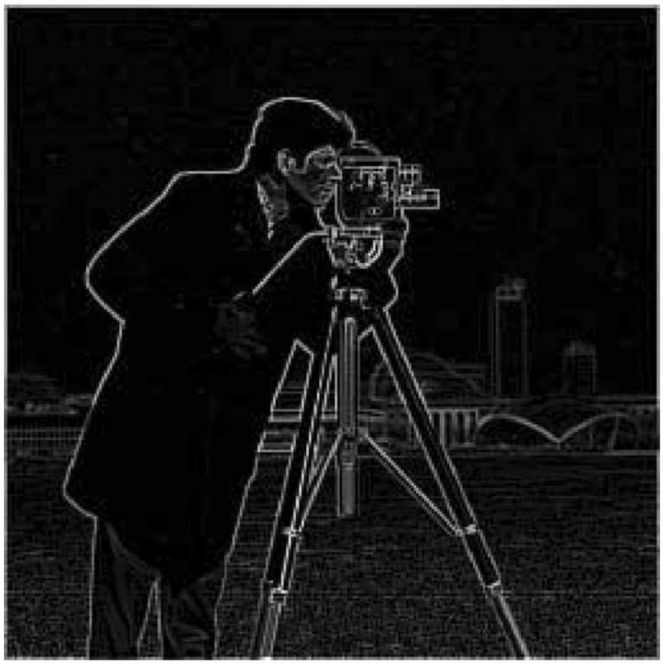
program formulated in Example 1. The initial state is loaded with ones except for the borders that are set to zero. Then, a one step program that produces an erosion of the state array and adds the input is iteratively applied. Note that the erosion eliminates layers of the state of the image, advancing one pixel per step. If the input image is empty, the state will converge to an empty image after a number of iterations that equals to the number of pixels in the largest dimension of the image. If there is a closed figure in the input, when the "erasing wave" reaches it, the program overwrites the contour by adding the input. Therefore, the closed figure in the state plane remains filled. The S-CNN function of the erosion and overlap can be combined in one simple program having the following F function :

$$x_5^{k+1} = x_5^k \cdot \left(x_1^k \cdot x_2^k \cdot x_3^k \cdot x_4^k \cdot x_6^k \cdot x_7^k \cdot x_8^k \cdot x_9^k\right) + u_5^k.$$

Example 3: Find the function F for the S-CNN program *edge detection*. This function is feedforward, and if the cell input value is zero it will remain as zero. If the value of the cell is one, it will remain as one only if at least one neighbor is zero. Even



a)



b)

Fig. 9. (a) Input image. (b) Output produced by the S-CNN edge detection program.

though the coefficients are programmed using 1 bit, the architecture can process grayscale images (Fig. 9) (see [25])

$$x_5^{k+1} = u_5^k \cdot \left(\overline{u_1^k} + \overline{u_2^k} + \overline{u_3^k} + \overline{u_4^k} + \overline{u_6^k} + \overline{u_7^k} + \overline{u_8^k} + \overline{u_9^k}\right).$$

A. S-CNN Library

In this section, we list a library of standard functions that can be used to perform image processing tasks. The functions can be combined in one S-CNN cycle or pipelined to accomplish the desired task in several steps. In this library the function F is described by the combination of the input mapping $G(U)$ and state mapping $F(X)$. Variables x_{i*j*} and u_{i*j*} correspond to the particular boundary conditions of the state and input, respectively, and x_{i0j0} represents the initial state (only specified when necessary).

- Edge Detection

$$G(U) = u_5 \cdot (\overline{u_1} + \overline{u_2} + \overline{u_3} + \overline{u_4} + \overline{u_6} + \overline{u_7} + \overline{u_8} + \overline{u_9})$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Corner Detection

$$\begin{aligned} G(U) = u_5 \cdot (&\overline{u_1 + u_2 + u_3 + u_4 + u_7} + \\ &+ \overline{u_1 + u_2 + u_3 + u_6 + u_9} + \\ &+ \overline{u_1 + u_4 + u_7 + u_8 + u_9} + \\ &+ \overline{u_3 + u_6 + u_7 + u_8 + u_9}) \end{aligned}$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Horizontal Translation

$$\text{Right}: G(U) = u_4$$
$$\text{Left}: G(U) = u_6$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Vertical Translation

$$\text{Up}: G(U) = u_8$$
$$\text{Down}: G(U) = u_2$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Diagonal Translation

$$\text{Up-Right}: G(U) = u_7$$
$$\text{Up-Left}: G(U) = u_9$$
$$\text{Down-Right}: G(U) = u_1$$
$$\text{Down-Left}: G(U) = u_3$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Point Extraction

$$G(U) = u_5 \cdot \overline{u_1 + u_2 + u_3 + u_4 + u_6 + u_7 + u_8 + u_9}$$
$$x_{i*j*} = 0, u_{i*j*} = 0.$$

- Point Removal

$$\begin{aligned} G(U) = &\text{Identity} - \\ &- (u_5 \cdot \overline{u_1 + u_2 + u_3 + u_4 + u_6 + u_7 + u_8 + u_9}) \end{aligned}$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Logic Not

$$G(U) = \overline{u_5}$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Directional Edge Detection

$$\text{Right}: G(U) = u_5 \cdot \overline{u_4}$$
$$\text{Left}: G(U) = u_5 \cdot \overline{u_6}$$
$$\text{Up}: G(U) = u_5 \cdot \overline{u_2}$$
$$\text{Down}: G(U) = u_5 \cdot \overline{u_8}$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Erosion

$$G(U) = u_1 \cdot u_2 \cdot u_3 \cdot u_4 \cdot u_5 \cdot u_6 \cdot u_7 \cdot u_8 \cdot u_9$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Dilation

$$G(U) = u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8 + u_9$$
$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Shadow Projection

$$\begin{aligned} \text{Right}: F(X) &= x_5 + x_6 \\ \text{Left}: F(X) &= x_5 + x_4 \\ \text{Up}: F(X) &= x_5 + x_8 \\ \text{Down}: F(X) &= x_5 + x_2 \\ x_{i*j*} = 0,\ u_{i*j*} &= 0. \end{aligned}$$

- Diagonal Line Detection

$$\begin{aligned} \text{Right}: G(U) &= u_1 + u_5 + u_9 \\ \text{Left}: G(U) &= u_3 + u_5 + u_7 \\ x_{i*j*} = 0,\ u_{i*j*} &= 0. \end{aligned}$$

- Global Connectivity Detection S-CNN

$$\begin{aligned} &\text{iterate } F(X) \cdot G(U)\ n \text{ times} \\ &G(U) = u_5 \\ &F(X) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \\ &x_{i*j*} = 0,\ u_{i*j*} = 0 \\ &x_{i0j0} = \text{one pixel of the image (load an image on the state that has one black pixel that belongs to the test image } u). \end{aligned}$$

- Hole Filling S-CNN

$$\begin{aligned} &iterate\ F(X) + G(U)\ n\ times \\ &G(U) = u_5 \\ &F(X) = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8 \cdot x_9 \\ &x_{i*j*} = 0,\ u_{i*j*} = 0 \\ &x_{i0j0} = 1. \end{aligned}$$

- Histogram Generation S-CNN

$$\begin{aligned} &\text{iterate } G(U)\ n \text{ times} \\ \text{Row}: G(U) &= u_4 \cdot u_5 + u_5 \cdot u_6 + u_6 \\ \text{Column}: G(U) &= u_2 \cdot u_5 + u_5 \cdot u_8 + u_8 \\ x_{i*j*} = 0,\ u_{i*j*} &= 0. \end{aligned}$$

- Contour Extraction S-CNN

$$\begin{aligned} G(U) &= u_5 \cdot Aux \\ Aux &= \overline{u_1} \cdot \overline{u_4} + \overline{u_1} \cdot \overline{u_2} + \overline{u_2} \cdot \overline{u_3} + \overline{u_3} \cdot \overline{u_6} + \\ &\quad + \overline{u_6} \cdot \overline{u_9} + \overline{u_9} \cdot \overline{u_8} + \overline{u_8} \cdot \overline{u_7} + \overline{u_7} \cdot \overline{u_4} \\ x_{i*j*} &= 0,\ u_{i*j*} = 0. \end{aligned}$$

- Gradient Detection S-CNN

$G(U)$

less than 4 ones in the neighborhood implies white, more than 6 ones in the neighbor implies white, black for the other cases

$$x_{i*j*} = 0,\ u_{i*j*} = 0.$$

- Logical AND S-CNN

$$\begin{aligned} &G(U) \cdot F(X) \\ &G(U) = u_5 \\ &F(X) = x_5 \\ &u \text{ Image 1} \\ &x \text{ Image 2} \\ &x_{i*j*} = 0,\ u_{i*j*} = 0. \end{aligned}$$

- Logical OR S-CNN

$$\begin{aligned} &G(U) + F(X) \\ &G(U) = u_5 \\ &F(X) = x_5 \\ &u \text{ Image 1} \\ &x \text{ Image 2} \\ &x_{i*j*} = 0,\ u_{i*j*} = 0. \end{aligned}$$

- Threshold

replace the cycle ramp value with the threshold value and run a complete cycle $x_{i*j*} = 0,\ u_{i*j*} = 0.$

V. Discussion

The CNN paradigm is based on a solid mathematical foundation and theory with a potential in image-processing applications. The proposed digital S-CNN architecture is *general* in the sense that the core cell can implement any Boolean function. In addition, the programming of tasks on the S-CNN platform is intuitive and can be done by writing down the truth table of the associated one-step logic function, as was illustrated in the examples in Section IV of the paper.

The architecture is also *scalable*, as a consequence of the broadcast mechanism used to communicate the information to all cells, i.e., the processing speed is independent of the size of the array. Intra-die communication scales well and even with todays technologies (3.3 V) digital signaling across the size of the whole array can be done in the many hundreds of Mbits/s at a cost of about 20 pJ per line. For an array with 1 million pixels, and a required throughput of frame rate of 16 ms, the overall cost for the ramp is approximately 20 mW. This is where most of the power will be dissipated in the system. The power for the single slope parallel A/D converter in the pixel is approximately the same i.e., 20 mW. The overall power dissipation for the whole system is approximately 40 mW. This is a very good number and it is between two to three orders of magnitude lower with multi-DSP or field-programmable gate array solutions that will be necessary to achieve the above throughput.

Perhaps, the most important aspect of the described S-CNN platform is that it provides a feasible digital realization of a CNN processor, with all the advantages of scaling, power and speed, which are typical of digital systems and Moore's law.

REFERENCES

[1] E. Fossum, "CMOS image sensors: Electronic camera on-a-chip," *IEEE Trans. Electron Devices*, vol. 44, pp. 1689–1698, Oct. 1997.

[2] C. Koch and H. Li, *Vision Chips: Implementing Vision Algorithms with Analog VLSI Circuits*. New York: IEEE Computer Society, 1995.

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

[4] T. Roska, A. Zarándy, S. Zöld, P. Földesy, and P. Szolgay, "The computational infraestructure of analogic CNN computing – Part I: The CNN-UM chip prototyping system," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 261–268, Jan. 1999.

[5] G. L. Nan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquez, "ACE4k: An analog I/O 64 $\times$ 64 visual microprocessor chip with 7-bit analog accuracy," *Int. J. Circuit Theory Appl.*, vol. 30, pp. 89–116, 2002.

[6] R. Dominguez-Castro, S. Espejo, A. Rodríguez-Vázquez, R. A. Carmona, P. Földesy, A. Zarándy, P. Szolgay, T. Szirányi, and T. Roska, "A 0.8-μm CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instruction storage," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1013–1026, July 1997.

[7] C. A. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, pp. 1629–1635, Oct. 1990.

[8] K. Boahen and A. Andreou, "A contrast sensitive silicon retina with reciprocal synapses," in *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann, 1992, vol. 4, pp. 764–772.

[9] K. Boahen, "The retinomorphic approach: Pixel-parallel adaptive amplification, filtering, and quantization," *Anal. Integr. Circuits Signal Processing*, vol. 13, pp. 53–68, 1997.

[10] T. Delbrück and C. A. Mead, "Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits," Caltech, Pasadena, CA, Tech. Rep., Caltech CNS Memo no. 30, 1996.

[11] K. A. B. E. Culurciello and R. Etienne-Cummings, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, pp. 281–294, Feb. 2003.

[12] C. Koch, "Seeing chips: Analog VLSI circuits for computer vision," *Neur. Comput.*, vol. 1, pp. 184–200, 1989.

[13] R. Etienne-Cummings, Z. Kalayjian, and C. Donghui, "A programmable focal-plane MIMD image processor chip," *IEEE J. Solid-State Circuits,*, vol. 36, pp. 64–73, Jan. 2001.

[14] A. Andreou and K. Boahen, "A 590 000 transistor, 48 000 pixel contrast sensitive, edge enhancing CMOS imager-silicon retina-," in *Proc. 16th Conf. Advaced Research VLSI*, Chapel Hill, NC, Mar. 1995, pp. 225–240.

[15] T. Serrano-Gotarredona, A. G. Andreou, and B. Linares-Barranco, "A programmable VLSI filter architecture for application in real time vision processing systems," *Int. J. Neural Syst.*, vol. 10, no. 3, pp. 179–190, 2000.

[16] D. Goldberg, G. Cauwenberghs, and A. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neutrons," *Neural Networks*, vol. 14, 2001.

[17] A. E. Gamal, D. Yang, and B. Fowler, "Pixel level processing — Why?, what?, and how?," in *Proc. SPIE Electronic Imaging '99 Conf.*, vol. 3650, San Jose, CA, 1999.

[18] S. Kleinfelder, S. Lin, X. Liu, and A. E. Gamal, "A 10 000 frames/s CMOS digital pixel sensor," *IEEE J. Solid-State Circuits*, vol. 36, pp. 2049–2059, Dec. 2001.

[19] F. Paillet, D. Mercier, and T. M. Bernard, "Second generation programmable artificial retina," in *Proc. IEEE Int. ASIC/SOC Con.*, Washington, DC, 1999, pp. 304–309.

[20] P. Julián, R. Dogaru, and L. O. Chua, "A piecewise-linear simplicial coupling cell for CNN gray-level image processing," *IEEE Trans. Circuits Syst. I*, vol. 49, pp. 904–913, July 2002.

[21] P. Julián, A. Desages, and O. Agamennoni, "High level canonical piecewise linear representation using a simplicial partition," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 463–480, Apr. 1999.

[22] L. O. Chua, "CNN: A vision of complexity," *Int. J. Bifurc. Chaos*, vol. 7, pp. 2219–2425, Oct. 1997.

[23] P. Julián, A. Desages, and B. D'Amico, "Orthonormal high level canonical PWL functions with applications to model reduction," *IEEE Trans. Circuits Syst. I*, vol. 47, pp. 702–712, May 2000.

[24] M. Chien and E. Kuh, "Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision," *IEEE Trans. Circuits Syst. I*, vol. CAS-24, pp. 305–317, June 1977.

[25] R. Dogaru, P. Julián, and L. O. Chua, "The simplicial neural cell and its mixed-signal circuit implementation: An efficient neural network architecture for intelligent signal processing in portable multimedia applications," *IEEE Trans. Neural Networks*, vol. 13, pp. 995–1008, July 2002.

Pablo Mandolesi (S'87–M'96) was born in Bahía Blanca, Argentina, on January 28, 1967. He received the Ingeniero Electrónico degree from the Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina, in 1991.

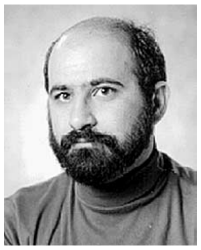
From June 2002 to July 2003, was a Visiting Scholar in the Sensory Communication Microsystems Laboratory, The Johns Hopkins University, Baltimore, MD. Since 1998, he is a Professor in the Electrical and Computer Engineering Department, UNS. His research interests are in the areas of system and circuit theory and design.

Mr. Mandolesi was president of the Argentinean chapter of the IEEE Circuits and Systems Society.

Pedro Julián (S'94–M'00) was born in Bahía Blanca, Argentina, on June 28,1970. He received the Ingeniero Electrónico degree and the Ph.D. degree in control de sistemas, from Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina, in 1994 and 1999, respectively.

From 2000 to 2002, he was a Visiting Scholar in the Nonlinear Electronics Laboratory, University of California at Berkeley. From 2002 to 2003, he was a Visiting Scholar in the Sensory Communication and Microsystems Laboratory, The Johns Hopkins University, Baltimore, MD. He has been an Assistant Professor in the Departamento de Ingeniera Elctrica y Computadoras (DIEC), UNS, since 2003. He also holds a position as an Assistant Researcher in the National Research Council of Argentina (CONICET) since 2002. His research interests areas are computation, circuits and systems theory and design. In particular, he is interested in practical and theoretical aspects of computation structures, including parallel systems like cellular neural networks and cellular automata, with a special emphasis in the design and analysis of very large-scale integration systems.

Dr. Julián serves as the Region 9 (Latin America) Vice President of the IEEE Circuits and Systems Society for 2004 and 2005.



Andreas G. Andreou (S'00–A'00) received the Ph.D. degree in electrical engineering and computer science from The Johns Hopkins University, Baltimore, MD, in 1986.

Between 1986 and 1989, he held a Post-Doctoral Fellowship and Associate Research Scientist position in the Electrical and Computer Engineering Department while he was also a Member of the professional staff at The Johns Hopkins Applied Physics Laboratory. He became an Assistant Professor of Electrical and Computer Engineering at the same university, in 1989, Associate Professor in 1993, and Professor in 1996. In 1995 and 1997, he was a Visiting Associate Professor and Visiting Professor, respectively, in the Computation and Neural Systems Program, California Institute of Technology, Pasadena. In the summer of 2001, he was a Visiting Professor at Tohoku University, Tohoku, Japan, working on three-dimensional integration. He is a member of the Whitaker Bioengineering Institute, The Johns Hopkins University, and Founding Director of the Whitaker Fabrication and Lithography facility. He is the co-founder of the Center for Language and Speech Processing, The Johns Hopkins University. His research interests include integrated circuits, sensory information processing, and neural computation. He is a co-editor of the book *Adaptive Resonance Theory Microchips* (Norwell, MA: Kluwer, 1998). He is Associate Editor of the journal *Neural Networks*.

Dr. Andreou is a recipient of the National Science Foundation Research Initiation Award and the 2000 IEEE Circuits and Systems (CAS) Society Darlington Award. He now serves as a Member of the Board of Governors for the IEEE CAS Society.