SCDVP: A Simplicial CNN Digital Visual Processor

Martin Di Federico, *Member, IEEE*, Pedro Julián, *Senior Member, IEEE*, and Pablo S. Mandolesi, *Member, IEEE*

***Abstract*—In this work we present a programmable and reconfigurable single instruction multiple data (SIMD) visual processor based on the S-CNN architecture, namely, the Simplicial CNN Digital Visual Processor (SCDVP), oriented to high-performance low-level image processing. The cells in the array have a selectable neighborhood configuration and several registers, which provide the chip with extended spatial and temporal processing capabilities, in particular optical flow. A prototype 64 × 64 cell chip with two program memories and a column adder was fabricated in a 90 nm technology, which running at 133 MHz delivers 105.5 GOPS. The calculation at the cell level is performed with time coded signals and the program memory is located outside the array. This produces a very efficient realization in terms of area: 53.8 GOPS per mm^2, which outperforms all results reported so far. We show that even after normalization, to account for technology scaling, the proposed architecture is the most efficient among all reported digital processors. Computation performance to power ratio also exceeds all previous results with 817.8 GOPS/W. Experimental results of the working chip are reported.**

***Index Terms*—Cellular neural networks (CNN), simplicial computation, image processing, pixel level processing, vision chip, ASIC, piecewise linear.**

I. Introduction

SINCE the early developments of image sensors, researchers have been interested in focal plane processors [1], [2]. With the advent of CMOS cameras a great variety of integrated systems has been proposed: A large number of vision chips have been developed with different architectures and processing paradigms [3]–[8]. They can be separated based on certain features such as: black and white or grayscale processing, analog or digital, SIMD or CNN, synchronous or asynchronous, application specific or general purpose, etc.

Mixed-signal vision chips integrate analog image processing in each pixel. Although analog vision chips are some times more power and area efficient than digital (see for example [4], [8]–[10]), the progress in CMOS fabrication process and the scalability of digital circuits have been increasing their advantages over analog designs. Digital vision chips are more flexible and permit the realization of more complex algorithms (see the references in [7]).

Manuscript received July 25, 2013; revised November 06, 2013; accepted November 24, 2013. Date of publication January 17, 2014; date of current version June 24, 2014. This work was partially supported by the International Project PICT 2010-2657 of ANPCyT of Argentina. This paper was recommended by Associate Editor Ricardo Carmona-Galán.

The authors are with Instituto de Investigaciones en Ingeniería Eléctrica, CONICET-UNS, Departamento de Ingeniería Eléctrica y de Computadoras (DIEC), Universidad Nacional del Sur (UNS), and Laboratory of Micro and Nano Electronics (LMNE) UNS-CIC; Av. Alem 1253, (8000) Bahía Blanca, Argentina.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2013.2295959

There are different approaches to allocate the processing elements (PE) to process an image. Architectures based on Single Instruction Multiple Data (SIMD) have a program stored in memory with instructions that each cell block executes. SIMD processors have quite dissimilar characteristics, which are a result of the particular application for which they were designed. If the circuit has a pixel sensor array (PSA), the PE can be located outside the PSA or embedded in the array. In the latter case, parallel processing can be performed at the expense of a larger pixel size. One example of this case is the cellular neural network (CNN) originally introduced by Chua [11].

In this paper, we are interested in general purpose digital visual processors with processing capabilities allocated for every pixel in the array. Several chips of these characteristics have been reported literature. The SRVC [5] is a SIMD prototype chip with an array of 16 × 16 PE and row/column processors to get coordinates. Every PE has limited computation capabilities consisting of a 4-bit register, a 2-input NOR and a 2-input NAND gate. The chip presented in [7] is programmable and based on multiple levels of parallel processors; it has a 128 × 128 PSA and a 32 × 128 PE array. The sizes of the PSA and the PE array (PEA) are different to reduce chip area. On the other hand, the PSA must output the image data column by column through 8-bit A/D converters and store the digital value into one of the columns of the PE. The iVisual [12] is composed of a 128 × 128 array of photosensors and reaches 1.16 GOPS/mm^2 but is based on row processors composed of 128 digital PE. The ASPA [6] combines asynchronous/synchronous processing techniques; this prototype chip has an array of 19 × 22 PE of 8-bit, connected in a rectangular grid. Every cell has seven 8-bit registers, an 8-bit multiplexed ALU and a bus controller. This provides more capabilities than [5] at the expense of area (the PE area is 100 μm × 117 μm in 0.35 μm technology and the fill factor is 2%). The Visual Attention Engine (VAE) presented in [13] implements a standard CNN (only linear CNN operations) with 4800 cells in 0.13 μm technology and reaches 5.298 GOPS/mm^2. In order to save area, the cells only contain 5 registers and the processing takes place in a processing cell which is shared by 40 cells; image acquisition is not performed on chip. Just recently, a mixed signal chip has been demonstrated at 100 kfps in [10]. The chip has an array of 256 × 256-32 μm × 32 μm pixel processors that operate on sampled-data current mode, and include in every cell 7 analog registers, 14 digital bits, a squarer, a comparator and bus control operations.

We propose a programmable SIMD visual processor using a simplicial piecewise-linear (PWL) CNN coupling cell [14], oriented to high performance low level image processing. The processor is intended to work with gray-scale level inputs. The proposed chip has a 64 × 64 array of cells, fabricated on a 90

1549-8328 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

nm CMOS technology. The proposed architecture is capable of spatial and temporal processing in fewer clock cycles than a previous implementation [15] with the addition of optical flow calculation. Every cell in the array implements a discrete-time nonlinear state equation, based on a multidimensional PWL function. The cells are programmed with a unique memory located on the periphery of the cell array, and the calculation is performed with time coded signals, which allows a very efficient realization in terms of area: with 53.8 GOPS/mm^2, it outperforms all reported visual processors. Each cell has three 6-bit registers and one 4-bit register. Two different functions can be computed at the same time using two of the registers as inputs, and a logical operation (AND, OR, XOR) can be applied to operate them. This allows the chip to make temporal processing, in particular, optical flow. In addition, two different neighborhood configurations can be selected providing more flexibility for spatial processing.

This work is organized as follows. Section II explains the chip architecture. Section III describes the experimental setup and elaborates on the obtained results. Finally, Section IV presents some conclusions.

A. Simplicial Cellular Nonlinear Networks

The core of the IC is an array of cells that computes locally according to the S-CNN paradigm [14]. Each cell is a nonlinear dynamical system that has an input $u_{i,j}(k)$ and a state $x_{i,j}(k)$ at a discrete time instant k and the cell state equation is given by:

$$\begin{cases} x_{i,j}(k+1) = F(\mathbf{x}_{S_{i,j}}(k)) \circ G(\mathbf{u}_{S_{i,j}}(k)), \\ y_{i,j}(k) = x_{i,j}(k), \end{cases} \tag{1}$$

where $i = 1,\ldots,N, j = 1,\ldots,M$; the cell functions $F : \Re^n \to \Re, G : \Re^n \to \Re$ are simplicial PWL functions; $\mathbf{x}_{S_{i,j}}(k) \in [0,1]^n$ and $\mathbf{u}_{S_{i,j}}(k) \in [0,1]^n$ are the state and input of the cell corresponding to a sphere of influence $S_{i,j}$; $y_{i,j}(k)$ is the cell output; n is the number of cells in the sphere of influence; N and M are the number of rows and columns of the array. Every component of vector $\mathbf{x}_{S_{i,j}}/\mathbf{u}_{S_{i,j}}$ is assumed to have a resolution of b bits (without loss of generality, every component is assumed to be normalized to the range [0, 1]).

This chip uses two PWL functions to determine the evolution of the cell. The next state of each cell is calculated according to the algorithm described in [14], which is briefly summarized next. The description is illustrated in Fig. 1, considering the parameters corresponding to the implemented case: $n = 5, b = 6$, i.e., five inputs per function, 6-bit precision each. The input $\mathbf{u}$ is a 5-dimensional vector formed by the input of the current cell, namely U_i, and the inputs of the four neighbor cells. The PWL function is uniquely defined by the 1-bit values assigned to the vertices of the hypercube $[0,1]^5$ (or $[0,63]^5$ in digital format), i.e., the 2^5 points $(0,0,0,0,0), (0,0,0,0,1), \ldots, (1,1,1,1,1)$. Therefore, the PWL function parameters are stored in a (1×2^5) memory that is addressed directly by the vertices. Inside the hypercube, the function must be interpolated to calculate its value. This requires to find the region (or simplex) where the point is located. For example, the (digital) point $\mathbf{u} = (5,1,40,10,51)$ belongs to the simplex with vertices $\mathbf{v_0} = (1,1,1,1,0), \mathbf{v_1} = (1,0,1,1,1), \mathbf{v_2} = (0,0,1,1,1), \mathbf{v_3} = (0,0,1,0,1), \mathbf{v_4} = (0,0,0,0,1), \mathbf{v_5} = (0,0,0,0,0)$ and in fact can be written as:

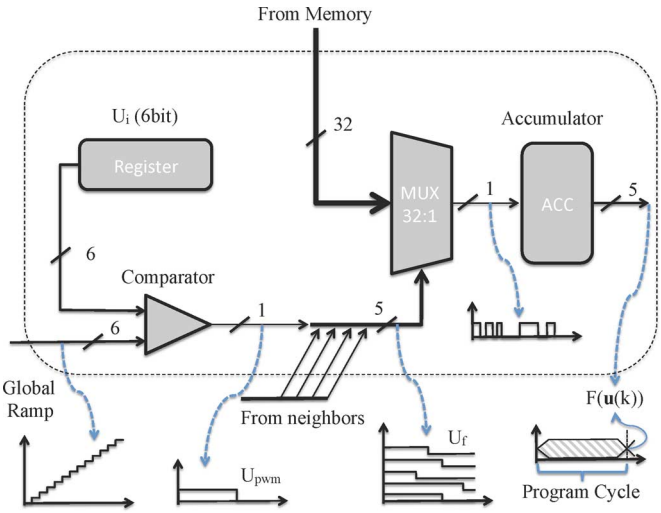


Fig. 1. Simplicial algorithm.

$\mathbf{u} = 1\times\mathbf{v_0}+4\times\mathbf{v_1}+5\times\mathbf{v_2}+30\times\mathbf{v_3}+11\times\mathbf{v_4}+12\times\mathbf{v_5}$. Inside every simplex the PWL function is linear, therefore $F(\mathbf{u}) = 1 \times F(\mathbf{v_0}) + 4 \times F(\mathbf{v_1}) + 5 \times F(\mathbf{v_2}) + 30 \times F(\mathbf{v_3}) + 11 \times F(\mathbf{v_4}) + 12 \times F(\mathbf{v_5})$. The function values are stored in the memory, for example, $F(\mathbf{v_3})$ is the content of the memory in location 00101. Inside the chip, this is calculated during a cycle, called *Program Cycle* (PC), of length 2^6, with the aid of a digital ramp as proposed in [14]. The 6-bit input of the current cell, U_i, is compared with the digital ramp, resulting in a 1-bit signal U_{pwm} coded in time. This signal and the other four corresponding to the neighbor cells are arranged to form a time-coded 5-bit word U_f. This word is used to address the memory and retrieve the function value $F(U_f)$. As all cells in the array share the same memory, the memory content is wired to all cells, and every cell makes a selection using a $(2^5 : 1)$ multiplexer (MUX). At every step of the PC, the value retrieved from the memory is integrated by an Accumulator (ACC). The number of clocks that one particular value from memory is accumulated is the coefficient multiplying that value. So, for the number $\mathbf{u} = (5,1,40,10,51), F(\mathbf{v_0})$ will be added one time, $F(\mathbf{v_1})$ will be added 4 times, $F(\mathbf{v_2})$ will be added 5 times, and so forth. At the end of the PC, the accumulator contains the value of the PWL function, just by adding the 1-bit memory contents.

This is a very area-efficient implementation of a nonlinear function since it only requires one comparator, one multiplexer and one accumulator per cell. The price for this simplicity is paid in time; one computation requires 2^6 clock cycles. If the next state of the cell depends only on one argument, this is all the computation needed. If it depends on both arguments, input and state, then two values, one for $G(U_f)$ and another for $F(X_f)$, must be retrieved from two memories and subsequently operated with a logical function (AND, OR, XOR) before being accumulated.

II. Chip Architecture

The architecture of the proposed chip is shown in Fig. 2. It is composed of a 64 × 64 array of interconnected cells. Outside the cell array there are sixty-four 12-bit adders, two 32-bit memories and control logic. The adder produces the sum of all cells

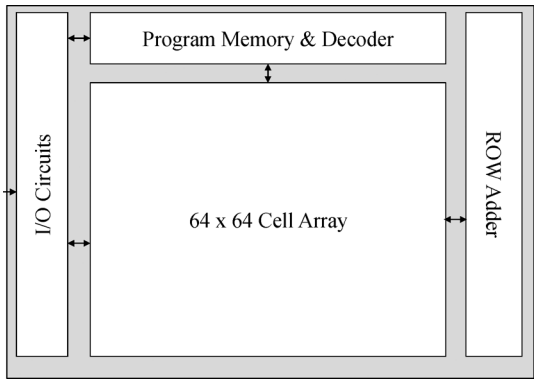


Fig. 2. Processor diagram block.

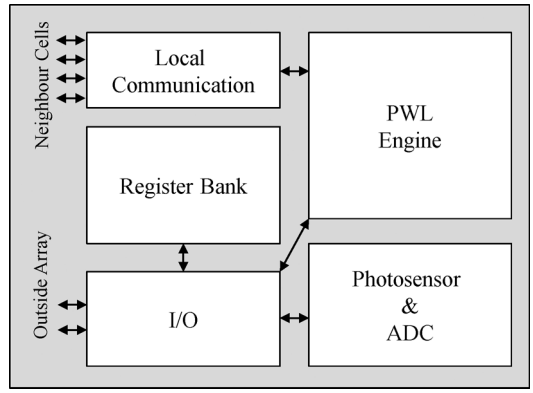


Fig. 3. Cell diagram block.

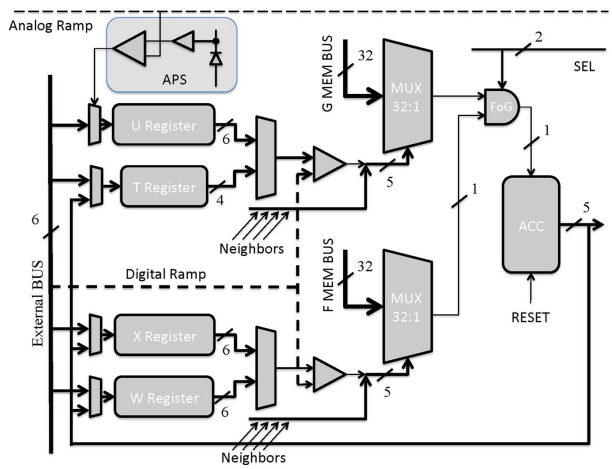


Fig. 4. Cell detailed architecture.

belonging to the same column and it has sixty-four 6-bit inputs (one for each row) and a 12-bit output.

In this case, the neighborhood is composed of the central cell and four adjacent cells, selectable between a vertical-horizontal $(+)$ or diagonal $(\times)$ configuration. Accordingly, it is possible to have two different neighborhoods of five cells that can be used to process using different templates. It is worth mentioning that the neighborhood size has a strong impact on the cell size [14] and also on the program memory length, which is proportional to the PC duration.

The memory, external to the array, called *Program Memory*, stores the 32-bit long instruction for the CNN processing task. This memory is implemented with two (*F* and *G*) 32-bit long serial registers and a 32-bit latch, which makes possible to load a function in the shift register while another is being used to process. As all cells use the *Program Memory* to get the function value for the processing task this value must be available to all cells. Therefore this memory is wired to all cells and a 32:1 multiplexer is located in each cell to fetch the memory information. As was explained in [16] the minimum cell (pixel) size is accomplished when 16 lines are used. As the architecture was intended to prioritize speed, the number of selected lines was 32. By doing this, the speed is doubled whereas the cell size is increased by 10 %.

Addressing circuitry enables random pixel access to read and write values in a particular cell register.

A. The Cell

Each cell has all the necessary circuits to perform the simplicial algorithm described in [14] and summarized on the introduction. The cell block diagram is shown in Fig. 3 and the detailed architecture is shown in Fig. 4. Each cell has a *Register Bank* to store information, a *PWL Engine* to process the neighbourhood information, a *Local Communication* block that communicate with the neighbours, an *Output Buffer* for external communication and an *APS and A/D converter* for image acquisition. These blocks are described next.

1) PWL Engine: The PWL Engine is the processing core of the cell. This block is comprised of a comparator, a 32:1 MUX, a Logic Unit and the ACC. The comparator is used to generate the PWM (U_{pwm}, X_{pwm}) signals used to process the time coded function. All the registers are multiplexed to use the same comparator, and the result of the comparison is stored in a pair of latches: one for the *F* function (FReg) and another for the

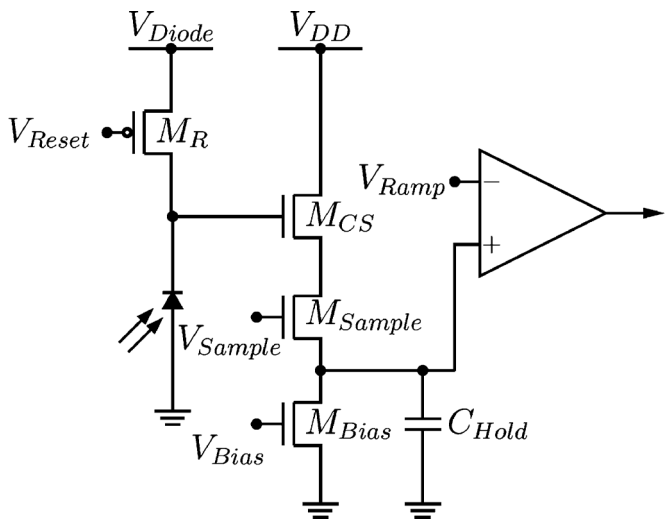


Fig. 5. Active Pixel Sensor schematic.

G function (GReg). This value is shared with adjacent cells to form the U_f and X_f words. The multiplexer is used to fetch the function value from the memory using the concatenation of all the PWM signals (U_f, X_f) belonging to the sphere of influence $x_{S_{i,j}}, u_{S_{i,j}}$. U_f and X_f are 5-bit long. The Logic Unit (FoG) is used to logically operate $G(U_f)$ and $F(X_f)$, to obtain a 1-bit output. The FoG block has 3 inputs: the two functions fetched from the memory $G(U_f)$ and $F(X_f)$, and the logical operation to perform. The ACC is used to integrate the output of the logic block.

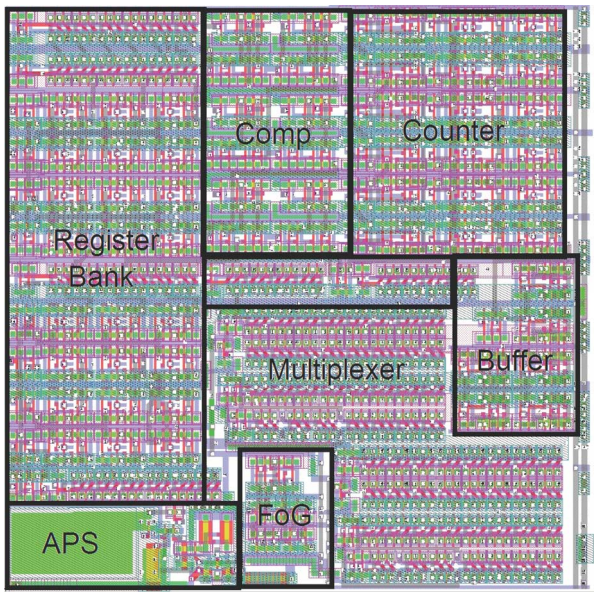


Fig. 6. Cell layout.

2) Register Bank: Each cell has four registers, three of them (X, U and W) have 6 bits and T has 4 bits. The three 6-bit registers are used to store images, and the T register is used either as a lower resolution image or to add flags or properties to the cell. The T register size allows to store the movement direction in the optical flow calculation (sixteen possible movement directions). This register can also be used as a function input, selectable between the external bus or the PWL Engine output. The U register stores the acquired image; X can store loaded or previously processed images; W is similar to X, but it can also be used for selective storage of values in certain cells. The T register stores only external values with selected enabling generated by the comparator.

3) Local Communication: The local communication connects the cell with the selected adjacent cells and sends the values needed to process. This block is comprised of a multiplexer that routes the connections to the ($+$ or $\times$) adjacent cells. These values (U_{pwm}, X_{pwm}) are concatenated to form the address of the simplex vertex (U_f and X_f) needed to calculate the function value. The F function is obtained from the address generated by the selected neighborhood (two sets of 5 cells, in a $+$ or $\times$ configuration). The G function can be fetched by the values provided by the 4 adjacent cells in a Horizontal-Vertical way ($+$) or from the T register (as the T Register is 4-bit, the most significant bit is set to '0'). By doing this, the cell can mask different regions of the image and apply different functions to different regions.

The areas of each cell block are indicated in Table I together with their power consumption per MHz. The cell size is $25\ \mu\text{m} \times 25\ \mu\text{m} = 625\ \mu\text{m}^2$; the PWL Engine occupies almost 50 % of the cell and is responsible for 57 % of the cell power consumption. Each cell has 1 294 transistors and the complete chip has 5 350 000 transistors. The layout of the cell is shown in Fig. 6.

III. Experimental Results

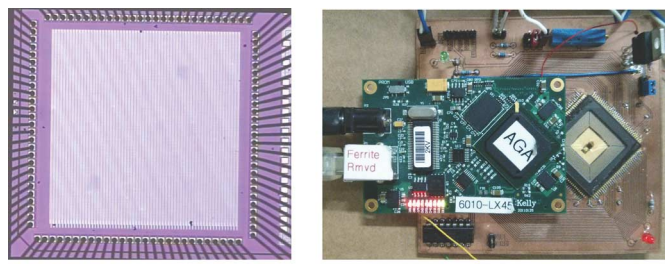
The prototype chip was designed and fabricated using the 90 nm one-poly, nine-metals CMOS process of the UMC foundry. The microphotograph of the chip is shown on Fig. 7(a). The full chip size is 2 mm $\times$ 2 mm, the working voltage for the core varies between 0.8 V and 1.2 V, and the I/O works from 0.8 V to 2 V. The chip has 114 pads, encapsulated in a CQFP128 package.

TABLE I
Cell Block Features

Block	Size	Power
FoG	12 μm^2	4.54 nW/MHz
Comparator	65 μm^2	24.6 nW/MHz
Counter	114 μm^2	43.2 nW/MHz
PWL Engine	360 μm^2	136 nW/MHz
Local Communication	12 μm^2	4.54 nW/MHz
APS	40 μm^2	
Output Buffer	33 μm^2	12.5 nW/MHz
Register Bank	185 μm^2	70.1 nW/MHz
Complete Cell	625 μm^2	236 nW/MHz

TABLE II
Chip Features

Parameter	Value
Technology	UMC 90nm 1P9M
Die Size	2 mm $\times$ 2 mm
Cell Size	25 μm $\times$ 25 μm
Fill Factor	2.5 %
Array Size	64 $\times$ 64
Number of transistors per pixel	1294
Photodiode Type	n+/p-Subtrate
Fixed Pattern Noise	8.08 %
Power Supply	1.1 V
Clock Frecuency tested	133 MHz
Power Consumption	129 mW



(a) (b)

Fig. 7. (a) Microphotograph of the chip; (b) Test board.

A board to test the prototype was fabricated. This board is connected to the bottom of an Opal Kelly 6010-l $\times$ 45 FPGA board using two surface mount 80-contact connectors (BTE-040-01), as displayed in the picture of the experimental setup (see Fig. 7(b)). The FPGA generates all signals to interface with the chip, including the clock signal. A PC running Matlab communicates with the FPGA through the USB port. The FPGA transmits the PC commands to the chip, reads signals from the chip, and then sends the information back to the PC.

In order to assess the maximum operation frequency, the clock frequency to the chip was increased until an incorrect chip output was observed. The test experiment consists in

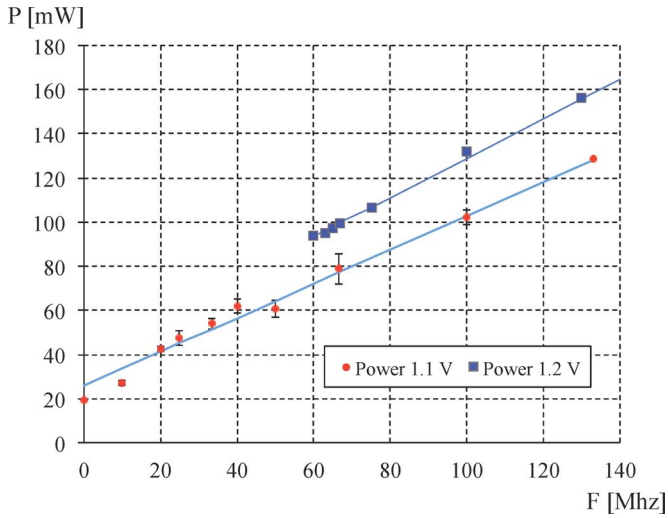


Fig. 8. Power consumption at $V = 1.1$ V and $V = 1.2$ V.

running a sequence of different operations at test speed, then reduce the speed and scan the whole array to check the result. The maximum frequency with correct operation was 133 MHz, although the chip was designed to operate at 200 MHz. In order to achieve the maximum operation frequency an internal PLL would be necessary. The limitation of 133 MHz is due to the board. An Agilent E5250 was used as power supply, which allowed to sense directly the average current consumed by the chip. Power consumption at 1.1 V and 133 MHz is 129 mW (17 mW of static power and 112 mW of dynamic power). Fig. 8 shows the curves of measured power consumption at 1.1 V and 1.2 V power supply versus frequency with the chip implementing eight different functions and a linear fit.

A. Experimental Image Processing

The chip works with different image resolutions, from 1-bit black/white images to 6-bit grayscale. The processing time depends on the data length. Black and white images are processed in 2/3 clock cycles, depending if the task uses 1/2 registers; similarly, 6-bit images are processed in 64/128 cycles. By default, the S-CNN algorithm implemented by the chip works with grayscale images, however some operations can be chosen to be performed just with black and white processing in order to speed up the execution.

This section shows experimental results of the chip performing different image processing tasks. The images in all cases were downloaded to the chip because the optical interface is currently under preparation. Measured fixed pattern noise is 8.08% so the use of synthetic images is convenient to work with good definition images.

1) Logical, Arithmetic and Morphological Operations: Logical operations can be performed with different images, with adjacent cell values of the same image and with operated images. Arithmetic operations can be done between different images. Morphological processing can be done directly with a $+$ or $\times$ structural element (SE), but it is also possible to simulate other SE by shifting the image. Figs. 9 and 10 show several examples.

2) Filters: Several filters can be applied with this structure. At the image level, we can mention: inversion, median, border detection, threshold, directional filtering, and several types of

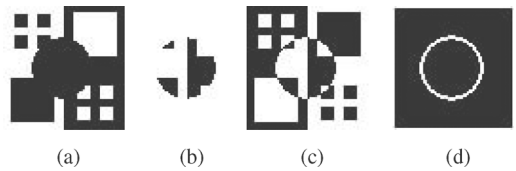


Fig. 9. Logical functions performed by the chip: (a) Circle AND squares; (b) Circle OR squares; (c) Circle XOR squares; (d) Circle XOR dilated circle.

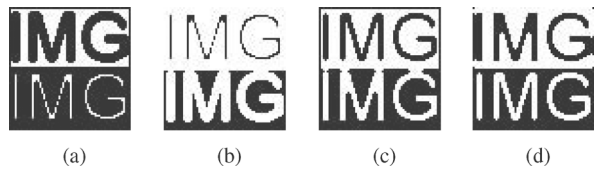


Fig. 10. Morphological operations performed by the chip: (a) Erosion; (b) Dilation; (c) Opening; (d) Closing.

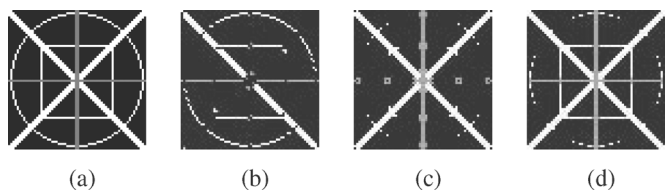


Fig. 11. Filters implemented by the chip: (a) Original image; (b) Horizontal and NW-SE diagonal lines filter; (c) Median filter with $\times$ neighborhood; (d) Median filter with $+$ neighborhood.

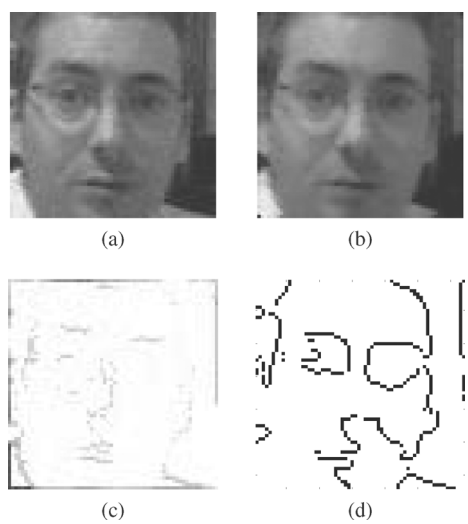
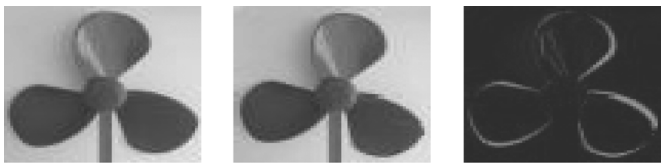


Fig. 12. (a) Original photo; (b) Closing; (c) Edge recognition; (d) Binarization applied to edge recognition.

border detection (horizontal, vertical, all borders, border with directionality, corners). Among line detection operations we can mention: vertical, horizontal, diagonal, curves, bifurcations, trifurcations, end of line. At the individual pixel level, isolated pixels can be detected and removed. Figs. 11 and 12 show several examples of filters used to process an image.

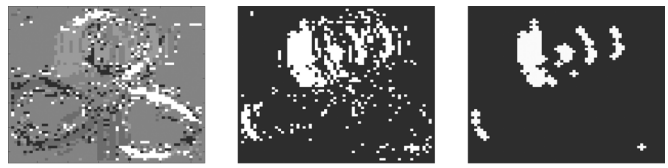
3) Correlated Double Sampling: The fixed pattern noise can be deleted by sampling the image, reset and subtract both images using the XOR function. This operation is performed at full image resolution.

4) Full Image Operations: Using the external adders it is possible to get the sum of all pixels in the image, by scanning all columns and integrating. In this case it is possible to detect a void image or measure the area/perimeter of an image by applying a sequence of operations. First of all, an edge detection



(a) (b) (c)

Fig. 13. (a) First image; (b) Second image; (c) Difference between images.



(a) (b) (c)

Fig. 14. (a) Optical flow calculation; (b) Pixels with a right movement; (c) Closing of optical flow selection to generate a mask.

is executed, followed by binarization with a given threshold. Then, if all pixels are added the perimeter is obtained. In order to obtain the area, after binarization a fill operation is performed, filling all closed figures. Then, if all pixels are added the area is obtained. This is useful, for example, if the area to perimeter ratio of an image, which is a standard image descriptor, has to be computed.

5) Transforms: The following transformations are possible: Hole filling, Global connectivity, Skeletonization, Bounding rectangle/diamond box, Centroid, Convex hull.

6) Pixel Coordinates: To get pixel coordinates, an image where each pixel has its own coordinates must be loaded and stored in the X and U registers, and then use the value to mask the image.

7) Range Detection: Using iterative directional dilation, it is possible to grow a bounding box that covers an object and extract the Range Detection.

8) Optical Flow: In order to calculate the optical flow, the current image is stored in U and the image at a previous instant is stored in X. Register W keeps the minimum value found for the difference between the current cell $\mathbf{u}_{S_{i,j}}(k)$ and any other neighbor cell at the previous instant $\mathbf{x}_{S_{i,j}}(k-1)$; W is initialized at the maximum possible value. Then, the following procedure is applied nine times (one per possible direction). The Bus outputs one direction, image X is shifted towards that direction, and the CNN performs the XOR between U and shifted X. This operation produces a difference image (pixel by pixel). Every cell compares this difference with W. If the calculated value is smaller than the one stored, W is updated with this new value and T is updated with the corresponding direction (available at the Bus). After nine steps, W has the smallest difference with all neighbors and T has the direction in which this holds. This calculation requires nine dual (U and X) PC, i.e., $9\times2\times2^6 = 1152$ clocks. At 133 MHz, the Optical Flow can execute at 115 451 fps.

Fig. 13 shows two consecutive images fed to the chip and the difference. Fig. 14(a) shows the Optical Flow calculated by the chip; white pixels are moving down, black pixels are moving up, dark gray pixels are moving left and light gray pixels are moving right. Fig. 14(b) shows the output of the function Right Movement Detection applied to the Optical Flow output, and Fig. 14(c) shows the Closing operation applied to the Right Movement Detection output, which deletes isolated pixels and detects the area of interest.

B. Performance and Comparison

In order to process one image, one comparison and one count must be done in each step of the PC. If two images are to be processed, then two comparisons, one logical operation and one count must be done. The first case requires one clock cycle, while the second requires two clock cycles.

One 64×64 image processed at 133 MHz requires two cell operations per clock (comparison and integration) for a total of $64\times64\times133\text{M}\times2 = 1.09$ TOPS. Two 64×64 image processed at 133 MHz requires four cell operations every two clocks (two comparisons, the logical operation and the integration), which also gives a total of 1.09 TOPS.

To process a 1-bit image takes 3 cycles. One for Reset, 1 for calculation and 1 to store the value in a register. Therefore, in three cycles the chip will perform the equivalent to six 1-bit multiplications and five additions (11 Arithmetic operations) plus the store and reset operations, for a total of 13 operations per cell. With the chip running at 133 MHz this is equivalent to $64 \times 64 \times 133\text{M} \times (13/3) = 2.36$ TOPS.

To process one 6-bit image takes 65 cycles. One for Reset, 63 for calculation and 1 to store the value in a register. Therefore, in 65 cycles the chip will perform a total of 13 operations per cell. With the chip running at 133 MHz this is equivalent to $64 \times 64 \times 133\text{M} \times (13/65) = 108.9$ GOPS. To process two 6-bit images takes 129 cycles. One for Reset, 127 for calculation and 1 to store the value in a register. Therefore, in 129 cycles the chip will perform the equivalent to 12 multiplications, 10 adds (22 Arithmetic operations), one logical operation between the images plus the store and reset operations, for a total of 25 operations per cell in 129 cycles. With the chip running at 133 MHz this is equivalent to $64 \times 64 \times 133\text{M} \times (25/129) = 105.5$ GOPS. At this clock speed, the chip can process 1 million 6-bit images per second. When the computational power is measured relative to silicon area, due to the time multiplexed computation, it achieves the best result reported in the literature so far: 53.8 GOPS per mm^2; more than seven times greater than the best result reported so far [10].

The operations per second to power ratio is 105.5 GOPS/0.129 W $=$ 817.8 GOPS/W which is the best result reported so far in the literature, improving that of [10].

Table III[1] shows the comparison of the proposed chip with other well known experimental chips. The row of PE OPS/#□ indicates the number of OPS achieved by a single PE divided by the technology-normalized PE area #□, which is defined as follows:

$$\#\square \triangleq \frac{\text{PE Area}}{(\text{feature size})^2} \tag{2}$$

This factor indicates the number of minimum size squares that fit into the PE area. Using this factor, the performance to

[1]The area corresponding to pads was subtracted prior to calculation for all chips.

TABLE III
REPORTED CHIPS CHARACTERISTICS AND PERFORMANCE COMPARISON

Parameter	This work	ACE16k [8]	SCAMP [4]	SRVC [5]	Parallel [7]	ASPA [6]	VAE [13]	Ref. [10]
Technology	90 nm	0.35 μm	0.35 μm	0.18 μm	0.18 μm	0.35 μm	0.13 μm	0.18 μm
Processing Type	Digital	Analog	Analog	Digital	Digital	Digital	Digital	Analog
Image Grayscale (bits)	6	$\sim$8	–	1	8	8	8	5.6
Memory per Cell (bits)	22	8ANA-2DIG	9ANA	4	72	64	40	7ANA-14DIG
Chip Size (no pads) (mm^2)	1.96	91	6.76	0.81	9.87	6.76	4.54	88.36
Cell Size (μm^2)	25×25	73.3×75.7	98.6×98.6	30×40	65×25	100×117	945[1]	32.26×32.26
Array Size	64×64	128×128	21×21	16×16	128×128	19×22	80×60	256×256
Number of PE	4 096	16 384	441	256	16 384	418	4 800	65 536
Transistors (M)	5.35	3.75	0.05	N/A	5.83	0.192	N/A	11.5
Frecuency	133 MHz	–	2.5 MHz	20 MHz	100 MHz	75 MHz	200 MHz	10 MHz
Performance (GOPS)	105.5	330	1.1	0.213	44	1	24	655
MOPS/mm^2	53 826	3 626	162	263	4 457	148	5 291	7 412
OPS/Transistor	19 719	88 000	19 487	N/A	7 553	5 200	N/A	56 786
PE OPS/#□	**333.8**	444	31	**22**	**53**	**25**	**89**	311
GOPS/W	817.8	82.5	27.5	24.4	97.8	37.8	285.7	532.5

area ratio is normalized with respect to technology in order to evaluate the processor efficiency regardless scaling. The factor is calculated for all chips even though it is only fully meaningful for digital designs (indicated in bold): the SCVDP shows the best PE OPS/#□ among the digital processors. A more relevant measure for analog/mixed-signal processors, included for completeness, is the number of operations per transistor: in this case, data show the analog processors outperforming the digital processors.

IV. CONCLUSION

We have presented a programmable and reconfigurable visual processor based on the CNN architecture. The processor can perform all CNN operations plus optical flow and mask operations. The cell neighborhood is configurable and the processor can be programmed using a single memory external to the array. A 64×64 proof-of-concept microchip was fabricated in a 90 nm technology, which was tested and is fully functional. Operating at 133 MHz, the chip delivers 105.5 GOPS. Due to the time multiplexed computation, the processor presents an excellent ratio of computational power to silicon area: 53.8 GOPS per mm^2 at 133 MHz, which is the best result reported in the literature by almost one order of magnitude. Moreover, once normalized, the proposed architecture turns out to be the most efficient among all reported digital processors, independently of technology, with 333.8 PE OPS/#□. In addition, it also features the best performance to power ratio reported so far, with 817.8 GOPS/W. Finally, we must stress that the structure can be scaled up in resolution/size without affecting speed or performance, and of course, being fully digital, it benefits directly from technology scaling.

ACKNOWLEDGMENT

The authors would like to thank Prof. Ming Du Ker, National Chiao Tung University (NCTU) and UMC for their support in the design and fabrication of the chip, Prof. Andreas G. Andreou from Johns Hopkins University for his cooperation in the testing environment, engineers Elian Dello Russo and Pablo Alvarez for their help throughout chip testing, and the anonymous reviewers who helped to considerably improve the manuscript thanks to their comments and suggestions.

REFERENCES

[1] C. Keast and C. Sodini, "A CCD/CMOS-based imager with integrated focal plane signal processing," *IEEE J. Solid-State Circuits*, vol. 28, no. 4, pp. 431–437, 1993.
[2] S. Kemeny, R. Panicacci, B. Pain, L. Matthies, and E. Fossum, "Multiresolution image sensor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 575–583, 1997.
[3] A. Andreou and K. Boahen, "A 590 000 transistor 48 000 pixel, contrast sensitive, edge enhancing, CMOS imager-silicon retina," in *Proc. 16th Conf. Adv. Res. VLSI*, 1995, pp. 225–240.
[4] P. Dudek and P. Hicks, "A general-purpose processor-per-pixel analog SIMD vision chip," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 1, pp. 13–20, 2005.
[5] W. Miao, Q. Lin, W. Zhang, and N.-J. Wu, "A programmable SIMD vision chip for real-time vision applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1470–1479, 2008.
[6] A. Lopich and P. Dudek, "A SIMD cellular processor array vision chip with asynchronous processing capabilities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2420–2431, 2011.
[7] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, 2011.
[8] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. Meana, "ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward vsocs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 5, pp. 851–863, May 2004.
[9] N. Massari and M. Gottardi, "A 100 db dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, 2007.
[10] S. Carey, A. Lopich, D. Barr, B. Wang, and P. Dudek, "A 100 000 fps vision sensor with embedded 535GOPS/W 256 × 256 SIMD processor array," in *Proc. Symp. VLSI Circuits (VLSIC)*, 2013, pp. C182–C183.
[11] L. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.
[12] C.-C. Cheng, C.-H. Lin, C.-T. Li, S. Chang, and L.-G. Chen, "Ivisual: An intelligent visual sensor SOC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," in *Proc. IEEE/ACM Design Autom. Conf.*, 2008, pp. 90–95.
[13] S. Lee, M. Kim, K. Kim, J.-Y. Kim, and H.-J. Yoo, "24-GOPS 4.5-mm^2 digital cellular neural network for rapid visual attention in an object-recognition SOC," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 64–73, 2011.

[14] P. Julian, R. Dogaru, and L. Chua, "A piecewise-linear simplicial coupling cell for CNN gray-level image processing," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 7, pp. 904–913, Jul. 2002.
[15] M. Di Federico, P. S. Mandolesi, P. Julian, and A. G. Andreou, "Experimental results of a simplicial CNN digital pixel processor," *Electron. Lett.*, vol. 44, no. 1, pp. 27–29, Dec. 2007.
[16] M. Di Federico, P. Julian, P. S. Mandolesi, and A. G. Andreou, "PWL cores for nonlinear array processing," in *Proc. 2010 IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2010, pp. 3312–3316.

Martín Di Federico (S'04–GS'07–M'12) was born in Bahia Blanca, Argentina, on August 17, 1981. He received his B.S. degree in electronic engineering in 2006 from the Universidad Nacional del Sur, Argentina, and his Ph.D. degree in engineering in 2011 from the same University. He was a BoG member of the Circuits and Systems Society and the actual Argentina Chapter Chair of CASS-IEEE. He visited the Sensory Communication & Microsystems Laboratory of the Johns Hopkins University and the Nanoelectronics and Gigascale Systems Laboratory at National Chiao Tung University. He is currently at the "Instituto Nacional en Tecnologia Industrial" INTI, Bahia Blanca, and Teacher Assistant in VLSI Circuits Design and Analog Circuits Design. Currently at INTI he is working in VLSI design and in Universidad Nacional del Sur in the development of CNN 3D pixel processors. His current field of research is the applications of VLSI circuits for image sensing and processing. His research interests include digital image processing, CNNs, neuromorphic circuits, and bio-electronic implants.

Pedro M. Julián (S'94–M'00–SM'05) received the electronic engineer degree in 1994 and the Ph.D. degree in control systems in 1999, both from Universidad Nacional del Sur (UNS), Bahia Blanca, Argentina. He was Visiting Scholar at UC Berkeley (2000 to 2002); Visiting Scholar (2002 to 2003) and visiting Fulbright Professor (2009) at Johns Hopkins University. He holds positions as an Associate Professor in the "Departamento de Ingeniería Eléctrica y Computadoras" (DIEC) at UNS, and as an Independent Researcher in the National Research Council of Argentina (CONICET). His research interests include theory and applications of computational circuits and systems, electronic systems, in particular sensory processors (acoustic and vision), with emphasis on low power VLSI systems. He served as the Region 9 (Latin America) Vice President and on the Board of Governors of the IEEE Circuits and Systems Society (CASS) from 2004–2007, and he is a founding member of the Latin American Consortium for Integrated Services (LACIS) and the Argentine School of Microelectronics (EAMTA). He is the recipient of the Bernardo Houssay 2009 Prize of the "Ministerio de Ciencia, Tecnología e Innovación Productiva" (MINCyT) and the 2009 Electronic Engineering Prize of "Academia Nacional de Ciencias Exactas, Físicas y Naturales" (ANCEFN). He also serves as Associate Editor of the *International Journal of Circuit Theory and Applications*. He is the principal investigator of "Tecnopolis del Sur" a hi-tech scientific park located in Bahía Blanca and Coronel Rosales.

Pablo Mandolesi (S'87–M'96) was born in Bahía Blanca, Argentina, on January 28, 1967. He received the "Ingeniero Electrónico" and Ph.D. degree in engineering from the Universidad Nacional del Sur, Bahía Blanca. From June 2002 to July 2003, was a Visiting Scholar in the Sensory Communication & Microsystems Laboratory of the Johns Hopkins University. Since 1998, he is a professor in the Electrical and Computer Engineering Department of the Universidad Nacional del Sur, Bahía Blanca, and he is a Researcher form the state of Buenos Aires research council "Comisión de Investigaciones Científicas" CIC. His research interests are in the areas of system and circuit theory and design. He was president of the Argentinean chapter of the IEEE-CAS.