INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS
Int. J. Circ. Theor. Appl. 2002; **30**:595–609 (DOI: 10.1002/cta.211)

A DSP-like analogue processing unit for smart image sensors

Antoine Dupret*,†, Jacques-Olivier Klein and Abdallah Nshare

Institut d'Electronique Fondamentale, UMR 8622, Bâtiment 220, Université de Paris XI, F91405 Orsay cedex, France

SUMMARY

An electronic retina featuring DSP-like programmable analogue processing is addressed. The motivations for designing such an original smart image sensor are accounted for. The architecture of the circuit is described and then the two more important building blocks are detailed. Finally, the practical implementation and tests results are given so as to validate the approach. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: analogue image processing; electronic retina; CMOS image sensor

1. MOTIVATIONS

To face the challenge of the image processing at video rate, numerous solutions were developed. The electronic retinas appear among one of the most prosperous axes of research [1–5]. Classically, the retinas are characterized by a completely parallel network of analogue operators that receive the signals directly stemming from photosensors. This total parallelism practically authorizes processing within a few milliseconds. Such an approach also offers very low power consumption along with a high operative density. Besides, the variety of the analogue operators, linear or not, is far richer than those of digital systems. Nevertheless, in real applications, the exploitation of such architectures remains marginal. We attribute this lack of interest to the three following difficulties.

First of all, for a given silicon area, finding a satisfactory compromise between the resolution and the completeness of the implemented analogue functions is a difficult task. This dilemma can either lead in two impasses. On the one hand, the retinas may offer a rich set of operators but at the expense of the resolution and the fill factor. Conversely, the circuit may feature an operative pan too much unadorned to afford exploitable results.

*Correspondence to: Dr. A. Dupret, Institut d'Electronique Fondamentale, UMR 8622, Bâtiment 220, Université de Paris XI, F91405 Orsay cedex, France.
†E-mail: antoine.dupret@ief.u-psud.fr

Published online 24 September 2002

Received 6 March 2001
Revised 9 November 2001

Secondly, the user of a retina is forced by the integrated operators foreseen by the designer. May a particular function be absent for an application and the use of a retina becomes impossible.

Finally, the interface of the retina is rarely taken into account from the very beginning of the design. Hence, their exploitation is afterward complicated.

Because of these three reasons, CMOS imagers or the CCD cameras associated to a digital, often programmable, system remain preferred to the retinas.

Following the example of the digital systems, we suggest using time as the new degree of freedom to go out of this impasse. In a digital processor, only some specific time-consuming functions are implemented in parallel. The other operations are decomposed into a sequence of basic instructions stemming from a reduced set of standard operators. These standard operators confer the numeric systems their so flexible programmable character. In this paper, we demonstrate that the digital approach may be successfully transposed to the electronic retinas using compact units of analogue processors. For it, we defined the architecture of a mixed digital–analogue processor containing an analogue computation unit. The circuit that we describe in this paper was designed to validate these standard analogue operators.

In the next sections, we begin by presenting the architecture of the circuit, and its two essential building blocks, and then the obtained experimental results.

2. ARCHITECTURE

2.1. Requirements for a programmable architecture

We first examined a set of algorithms liable to be implemented on smart image sensors. Mainly, smart sensors are designed to reduce the amount of data so that only the relevant ones are kept. Hence, non-linear processing like binary mathematical morphology, neural [6] or CNN [5,7–10] processing, for edge/motion detection and image segmentation is required. In addition, artificial retinas deal with arrays of raw analogue values and have to perform low-level image processing: linear or non-linear filtering for image regularization.

From this observation, we have listed the primitives needed for these algorithms. On the one hand, two-dimensional convolution is used for linear filtering. On the other hand, edge detection and motion detection require comparison and thresholding while neural networks or CNN require non-linear functions and mathematical morphology implies the use of Boolean operators.

Finally, we have derived a list of functional units for these primitives.

An analogue processor is dedicated to linear operations. It is able to execute a multiply-accumulate operation (MAC):

$$A \leftarrow A + c \times D$$

where c is a signed fixed point coefficient less than one, A is an accumulator and D is a datum. Since signed operations simplify the implementation of practical algorithms, the subtraction comes along. The non-linear operations are

- the comparison:

$$A \leftarrow D < V_{\text{ref}}$$

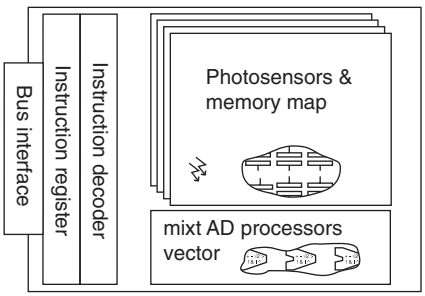

Figure 1. PARIS chip bloc diagram.

- five Boolean operations:

$$A \leftarrow A$$
$$A \leftarrow A \times B$$
$$A \leftarrow A \times \bar{B}$$
$$A \leftarrow A + B$$
$$A \leftarrow A \oplus B$$

- and the conditional execution of an analogue instruction, with a local condition resulting of a comparison or a Boolean operation.

In the test version here described, the Boolean operators and the conditional instruction have not been integrated.

2.2. *Cellular architecture*

The architecture articulates around four blocks: an array of 16×16 pixels, a 16 lines vector of processors, an instruction decoder and a bus interface (cf. Figure 1). The array of pixels is constituted of cells, each one containing a photosensor and four analogue memories. The stored data can be Boolean or in continuous values. The array is designed so as to allow a random access, in read or write mode, of one of the four analogue registers of one of the pixels of every row. Every pixels of the same row share the same bus called *analogue–digital bus* (ADB).

A processor connects with every row of pixel. The processing of resulting data of the various registers are performed on these processors. A 3-to-1 multiplexer is set between the row of pixels and the processor (cf. Figure 2). Therefore, a datum on the ADB of the current row or of one of the two nearest rows can be processed. The analogue computation unit stores the result of the computation in an analogue memory playing the role of the accumulator. The control of the retina is single instruction multiple data (SIMD). So at a given time t, all the processors execute the same instruction. The set of instructions of the analogue computation unit is an operation between the contents of a register and the accumulator. The transfers from one register to another register or from the accumulator to a register were also implemented.

To ease the evaluation when the retina is implemented in a system, PARIS can be seen as a standard peripheral of a microcontroller unit (MCU). So the interface bus comprises the set

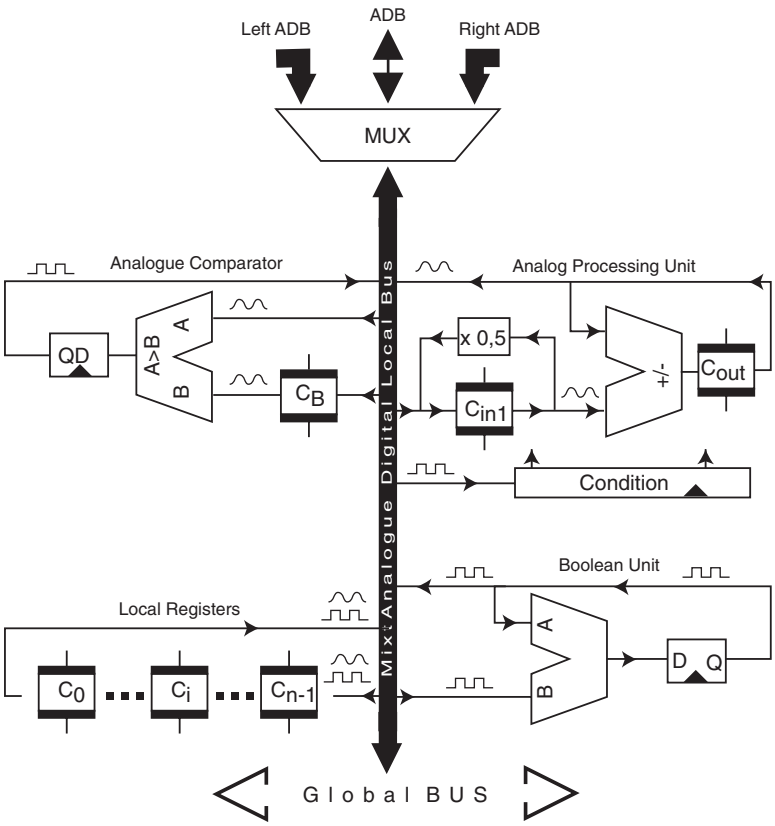

Figure 2. Functional architecture of a mixed AD processor.

of standard bus signals ($\overline{RESET}, \overline{CS}, \overline{RD}, \overline{WR}$, A[1:0], D[7:0]). The external MCU writes the control bytes into the instruction register of PARIS, which is constituted of 48-bit registers. During a write cycle, the corresponding instruction is latched, decoded in a combinatorial fashion and executed (i.e. generate the actual control signal of the analogue part).

In addition to the optical input, an analogue input/output connected with an external DAC/ADC has also been added. The voltage on this input can be driven anywhere in the analogue part of the circuit. It is therefore possible to upload an electronic image during the tests. The test and the characterization are thus simplified.

Finally, our ultimate objective is to demonstrate the feasibility of our approach for a 256×256 pixel retina with a silicon area of $1/4\ \text{in}^2$. The main elements we now describe were designed in that perspective.

2.3. *Pixel*

Since our goal is to integrate a 256×256 pixel retina on a $1/4\ \text{in}^2$ I.C., the maximum pixel area is $50 \times 50\ \mu\text{m}^2$. The pixel (cf. Figure 4) is organized around a photosensor and four storage capacitances. The transfer of the data from the bus ADB to one of these memories

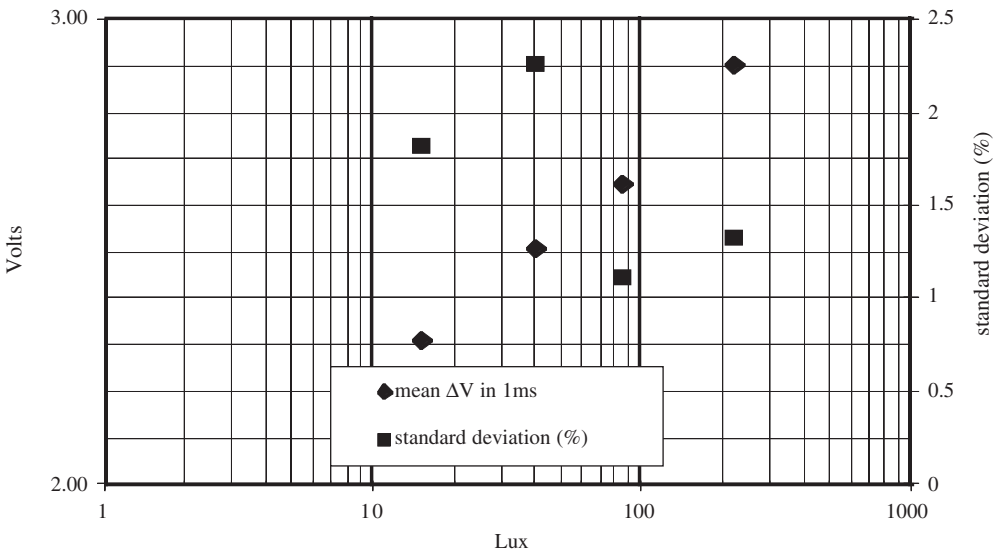

Figure 3. Measured sensibility and fixed-pattern noise as a function of ambient brightness.

or from a capacitor to the ADB is achieved thanks to a bidirectional buffer and a set of switches.

Minimal size CMOS switches are used in order to reduce the effect of stray capacitances and charge injection. Now, the value of the storage capacitors result of a tradeoff between the area, the write pulse width and the accepted error introduced by charge redistribution. We have adopted 280 fF NMOS capacitors featuring a high capacitance per surface unit.

The same constrains apply to the buffer but, in addition, its dynamic must be taken into account. Hence, to keep the voltage follower small, it is constituted of a five-transistor OTA. The static gain of the OTA derives from the bearable error of copy while the access read/write time requirements determines its Gain Band Width product and slew rate.

Of course, the input transistors of the OTA lead in parasitic capacitances, which have to be taken into account for the sizing of the storage capacitors. However, these parasitic capacitances constitute the main error source. Thus, for future design, we are investigating others solutions, derived from Reference [11], less sensitive to parasitic capacitances.

To handle all the data of the pixels in a homogeneous way, the photosensor works in integration mode, discharging one of the capacitors beforehand preset to a voltage V_0.

The remaining available pixel area is limited (11% of $50 \times 50 \mu m^2$) yet the exposure duration must remain below 20 ms under typical daylight lighting (corresponding to a minimum optical flow of $3\,mW/m^2$). Since the 280 fF storage capacitance must feature a discharge of 3 V the sensibility have to be greater than 5 A/W. On our CMOS technology, photodiodes are limited to about 0.1 A/W; thus the photocurrent has to be amplified by a factor of at least 50. We so chose to use vertical PNP transistors as phototransistors. For an exposure duration of 1 ms, the measured discharge of the storage capacitance and the fixed-pattern noise (FPN) as a function of the ambient brightness is given in Figure 3. Vertical PNP transistors introduce an FPN around 2% (results close to those of Reference [12]) which should be reduced thanks

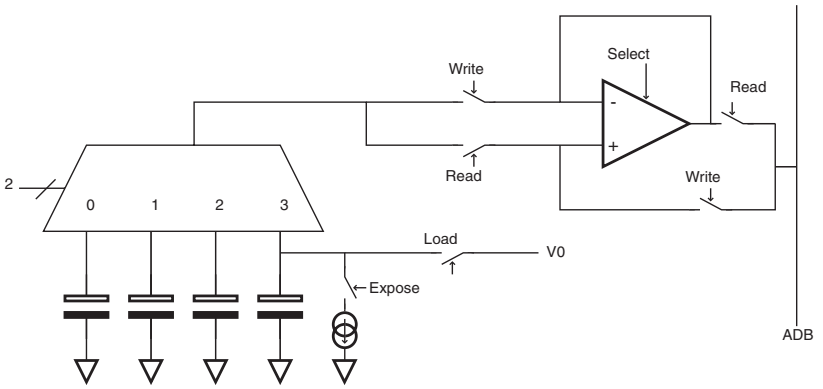

Figure 4. Simplified schematics of the pixel.

Table I. OTA characteristics.

Characteristics	Value
Gain bandwidth	15 MHz
Power consumption	50 μW
Static gain	> 54 dB

to an appropriate technique [1,13]. However, this point has not been addressed in our design (Figure 4).

Since the circuit is designed so that its cells can be directly reused in a 256×256 retina, the silicon area of the voltage follower and its power consumption have to be the lowest possible. In this perspective, the OTA are turned into idle mode when the corresponding pixels are not addressed. This technique does not affect significantly the access time because the startup time of the voltage follower is about 10 ns. The static gain of the OTA derives from the bearable error of copy while the access read/write time requirements determines its gain band width product and slew rate. Simulations on typical image processing algorithms have lead to specify a static gain around 1000, a minimum GBW of 10 MHz, and a slew rate of 40 V/μs. The characteristics of the OTA are summarized in Table I.

2.4. Programmable analogue–digital processor

The programmable analogue–digital processor includes an analogue computing unit, a comparator and an analogue register (cf. Figure 2). Indeed, so as to reduce the total silicon area of the circuit, the Boolean operators—which are just constituted of standard cells and do not include original concept—have not been integrated in this chip (Table II).

As previously stated, the computations are performed in a sequential way by the processors. The chain of instructions imposes that the device works in discrete time, holding data during one clock period. Then, the key point is to be able to design a building block able to achieve successions of MAC. We so retained as a basic scheme a structure inspired of a switched capacitors integrator. The schematic is given in Figure 5.

Table II. Main characteristics of the evaluation circuit.

Characteristics	Values
Total area of the circuit	10 mm^2
Resolution (pixel)	16×16
Number of processing unit	16
Area per processing unit	50×200 μm^2
Area per pixel	50×50 μm^2
Max. clock frequency	10 MHz
Power consumption per pixel	100 μW
Power consumption per processing unit	300 μW

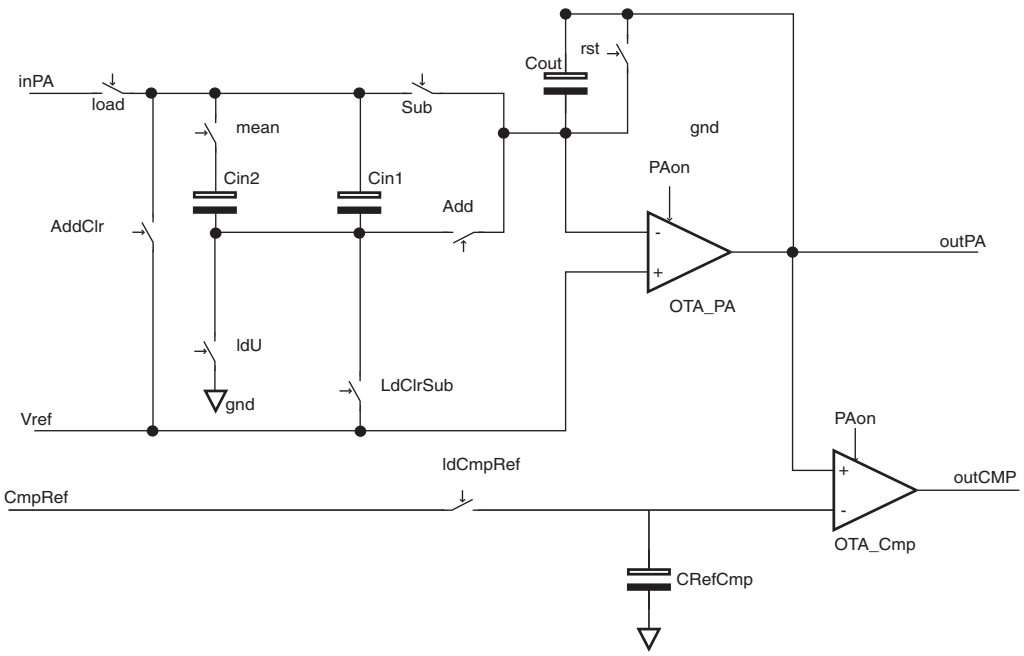

Figure 5. Analogue processing unit.

The capacitance C_{out} plays the same role as the accumulator in a digital processor. The charge, loaded in C_{in1}, is transferred to C_{out}. According to the switches *Add* and *Sub* the charge from C_{in1} adds or subtracts to the charge in C_{out}, thus implementing the addition and the subtraction. The multiplication by a constant consists in applying a voltage V_{in} to the capacitor C_{in1} while C_{in2} is reset. Next, C_{in1} and C_{in2} are connected together. Since C_{in1} and C_{in2} are of equal value, the charge in C_{in1} is half its previous value. Iterating i times this step leads to a charge in C_{in1} of the form

$$Q_{in1} = C_{in1} \times V_{in} \times 2^{-i}$$

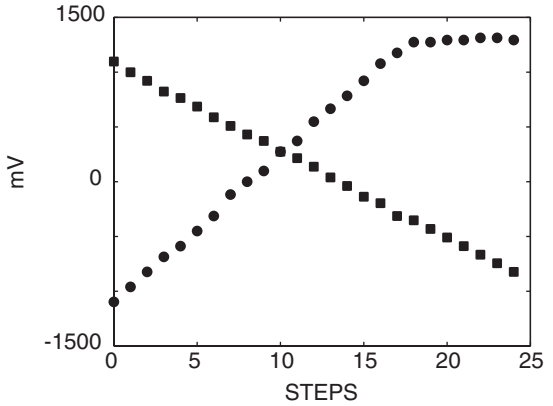

Figure 6. Generation of two ramps by successive additions/subtractions operations.

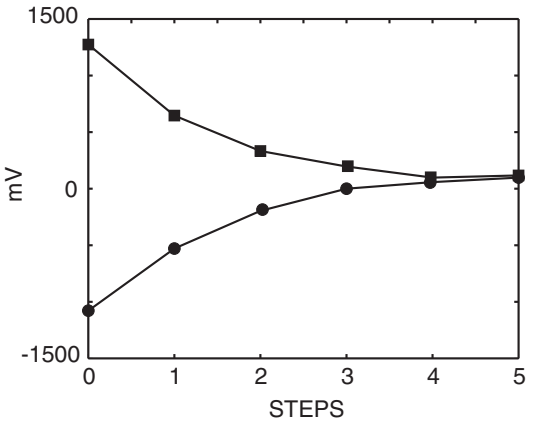

Figure 7. Generation of two exponents by successive 'divide by two' operations.

The remaining charge in C_{in1} is then transferred to C_{out} and adds with the charge in it. Finally, when this stage is iterated N times, the voltage at the output of the OTA is

$$V_{\text{out}} = V_{\text{in}} \sum_{i=1}^{N} a_i \times 2^{-i}$$

It is therefore possible to multiply any voltage by virtually any constant value lower than 1. The successive products being added in the accumulator, the MAC is achieved. An extra set of switches allows to perform signed or unsigned operations.

The specifications of the OTA in the analogue processor are noticeably identical to those of the follower used in the pixels. The capacitors are Poly1/Poly2 capacitances of identical layout, placed one next to the other to reduce the relative dispersals.

The capacitance value of C_{in1}, C_{in2}, and C_{out} has been determined so that the charge injection resulting from minimum size switches introduces an output error of a few millivolts [14]. This

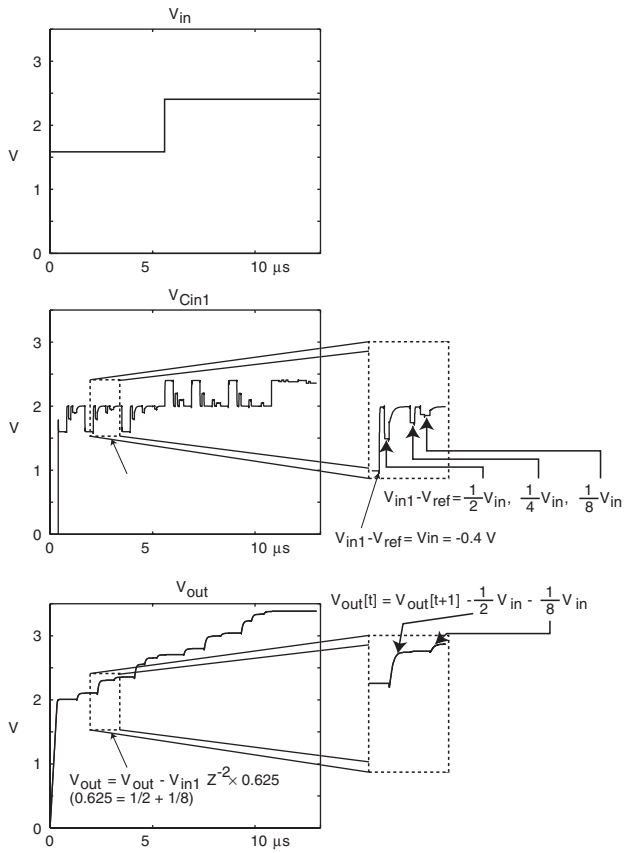

Figure 8. Layout extracted simulation of the analogue unit computing the step response of an FIR filter (with impulse response $h(z) = 0.25z^{+3} + 0.625z^{+2} + 0.875z^{+1} + 0z^{-0} - 0.875z^{-1} - 0.625z^{-2} - 0.25z^{-3}$).

voltage is of the same order of magnitude as the one derived from the static gain of the OTA. Hence, a simple and compact switching strategy can be retained.

To demonstrate the actual working of the processor, measurements corresponding to a succession of subtractions, of additions then of divisions by 2 are given (Figures 6 and 7).

The comparator samples the voltage on the global bus and stores it in a capacitor. The voltage of this capacitor and of the output of the analogue processing unit are then compared thanks to an OTA. The result of the comparison may be set on the local bus of the digital–analogue unit.

An analogue register was added to the processor. It allows storing an intermediary result without having to rewrite it in one of the memory of the array.

Finally, a set of multiplexer enables to connect a processor to the *ADB* of the pixels of its row or the *ADB* of the following or of the previous row. It is so possible to implement filters with a $3 \times N$-pixel convolution kernel. As an example, Figure 8 represents the computation of a Canny ID filter [15].

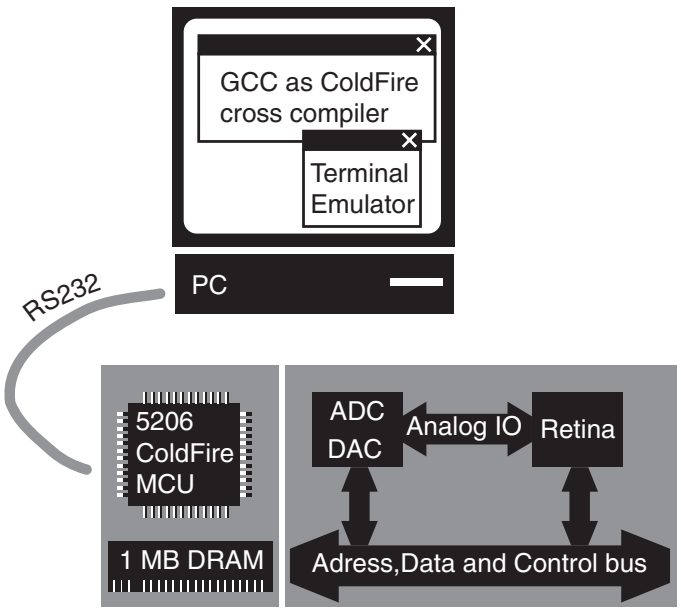

Figure 9. Hardware architecture of the test system.

3. SYSTEM IMPLEMENTATION

3.1. Hardware and software environment

In this section, the description of the system is addressed. The MCU handles the retina that constitutes one of its memory mapped peripherals (cf. Section 2.2). The MCU plays the role of a sequencer providing the succession of instructions corresponding to the implemented algorithm. Once the computation is completed on PARIS, the data are downloaded to the PC via the MCU. Finally, the results can be displayed and stored on the PC. Obviously, the PC is convenient simply during the development stage (Figure 9).

The algorithms are written in C++ language. The binary code is produced by a cross-compiler hosted on the PC with the MCU as target. The software architecture is organized as follows:

Firstly, each retina micro-instructions is defined as one constant to be written in one of the bit-field of one of the registers of PARIS. Secondly, micro-instructions are grouped in functions to constitute the 28-macro-instruction set (cf. Figure 10). Finally, the algorithms are written calling the macro-instructions.

Such a software/hardware environment has proven to be an efficient way to test, debug and finally implement algorithms on the retina.

3.2. Example of an implemented filter

To illustrate how algorithms are invoked, we present the implementation of a zero crossing of the Laplacian edge detector. The convolution kernel K of the zero crossing of the Laplacian


```
//Types definition.
typedef unsigned char U8;
typedef volatile U8 & REG8;

//Examples of constants definition for Analog Computation Unit micro-instructions.
typedef enum  {
 nop=0,
 ClrOut=1,
 LoadIn=2,
 SemiClr=3,
 Move_Add=4,
 Move_Sub=5,
 Div_2=6,
 LoadCmp=7,
 RegWE=8,
 PAOff=9,
 LoadInU=10
}  CodeOp_PA_T;
...
//4 8-bit registers definition and initialization
REG8 CODE_OP = (REG8) *(U8*) (PARIS3_ADD + 0);
REG8 AD_MODE = (REG8) *(U8*) (PARIS3_ADD + 1);
REG8 LI_ADDR = (REG8) *(U8*) (PARIS3_ADD + 2);
REG8 CO_ADDR = (REG8) *(U8*) (PARIS3_ADD + 3);
...
//Examples of macro-instructions definition.
inline void add (void){
CODE_OP = Move_Add ;
CODE_OP = nop;
}
inline void sub (void){
CODE_OP = Move_Sub ;
CODE_OP = nop;
}
inline void div_by_2 (void){
CODE_OP = SemiClr ;// reset of Cin2
CODE_OP = nop;
CODE_OP = Div_2 ;// Charge redistribution between Cin1 and Cin2
CODE_OP = nop;
}
inline void pixel_to_pa (U8 l, U8 k){
AD_MODE = current | MX_K*k;
LI_ADDR = l |0x80;
CODE_OP = Read;
CODE_OP = Read | LoadIn ;
CODE_OP = Read  ;
CODE_OP = nop;
LI_ADDR = 0;
AD_MODE = 0;
}
...
```

Figure 10. Software examples.

edge detector is the following one:

$$K = \begin{bmatrix} 0 & -\frac{1}{4} & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{bmatrix}$$

The following C++ source code implements the convolution of an image with the kernel K. It operates on vectors (rows of ARAM, V_{in} and V_{out}, and iterates on all rows:

```
void edge_detect(U8 ksrc) {          // ksrc is the memory map index
   int r;                            // row index
```


Figure 11. Original image uploaded to the retina.

```
for(r=1;r<15;r++){                  // iterate on rows 1 to 15
     clear();                       // Cout reset <=> Vout = 0
     pixel_to_pa(r-1,ksrc);         // load previous row as Vin
           div_by_2();              //     \
           div_by_2();              //     }Vout = Vout - 1/4 Vin
           sub();                   //     /
     pixel_to_pa(r+1,ksrc);         // load next row as Vin
           div_by_2();              //     \
           div_by_2();              //     }Vout (t+1) = Vout (t) - 1/4 Vin
           sub();                   //     /
     pixelnext_to_pa (r,ksrc);      // load current row shifted left as Vin
          div_by_2();               //     \
          div_by_2();               //     }Vout (t+1) = Vout (t) - 1/4 Vin
          sub();                    //     /
     pixelprevious_to_pa (r,ksrc); // load current row shifted right as Vin
          div_by_2();               //     \
          div_by_2();               //     }Vout (t+1) = Vout (t) - 1/4 Vin
          sub();           //       /
  pixel_to_pa(r,ksrc);              //     load current row as Vin
       add();                       // Vout (t+1) = Vout (t) + Vin
 }
```

Finally, Figures 11–13 represent the results of the filtering, using the kernel K, of an image on PARIS.

Since linear and non-linear functions are liable to be implemented, CNN algorithms can be simply programmed. Examples of CNN implementation have been described in Reference [16]. Table III gives the time requirements for typical CNN operations.

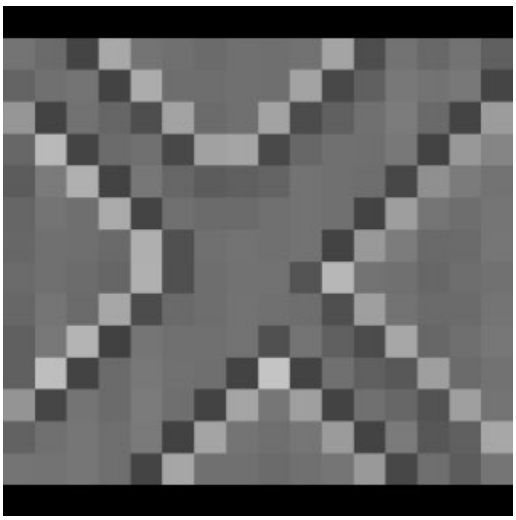

Figure 12. Convolution result of the previous image with the kernel K (out of the analogue processors).

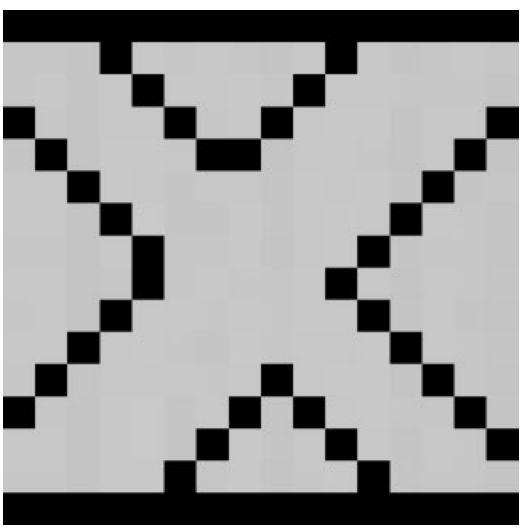

Figure 13. Thresholding result of the previous image (out of the comparator).

4. CONCLUSION

A programmable smart image sensor has been presented. The main characteristics of the circuit are summarized in Table II. So as to demonstrate the feasibility of the chosen approach, 16×16 pixels circuit has been realized using a 0.6 μm DLM, DLP CMOS technology (a microphotography of PARIS is given Figure 14). In particular, the tests of the processors allow to envisage efficient image processing at a video rate. With pixels having a silicon

Table III. Time requirements for typical CNN operations.

Functions	Instructions	Duration
Temporal integration	5	0.5 μs
Non-linear funct (3 seg.)	20	2 μs
Others (DC offset, loop ctrl…)	25	2.5 μs
Total for one row	100	10 μs
Total for 128 rows	12 800	1.28 ms

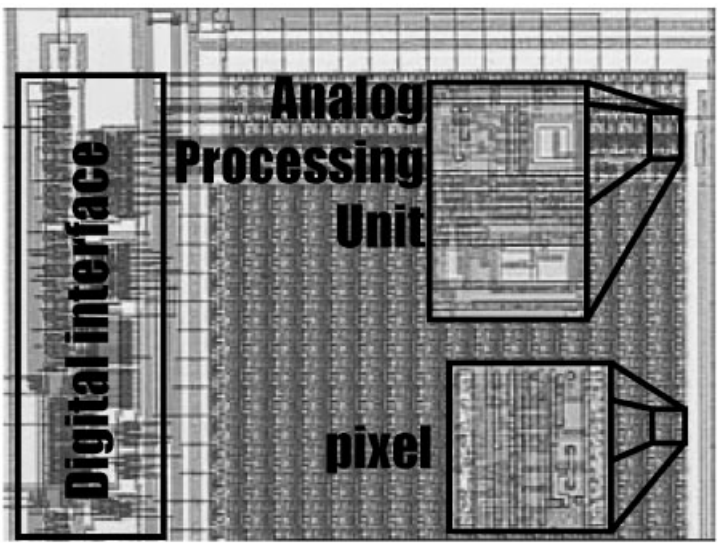

Figure 14. Microphotographie of PARIS.

area of $50 \times 50\,\mu m^2$, a 256×256 pixel retina can be sent to foundry. The silicon area of the programmable analogue–digital processor is small enough not to add significant surface on large retina. A new version of this architecture will soon implement conditional operations.

REFERENCES

1. Mead CA. *Analog VLSI and Neural Systems*. Addison-Wesley: Reading, MA, 1989.
2. Moini A. *Vision Chips*. Kluwer Academic: Dordrecht, 1999.
3. Kyuma K, Lange E, Ohta J, Hermanns A, Banish B, Oita M. Artificial retinas-fast, versatile image processors. *Nature* 1994; **372**:197–198.
4. Bernard T, Zavidovique B, Devos F. A programmable artificial retina. *IEEE Journal of Solid State Circuits* 1993; **28**:789–797.
5. Roska T, Rodriguez-Vazquez A. Toward the visual microprocessor. *VLSI Design and the Use of Cellular Neural Network* (*CNN*) *Universal Machine Computer*. Wiley: New York, 2001.
6. Graf HP, Sackinger E, Boser B, Jackel LD. Recent developments of electronic neural nets in US and Canada. *Proceedings of the Second International Conference on Microelectronics for Neural Networks*, Munich; 471–488.
7. Chua LO, Yang L. Cellular neural networks: theory. *IEEE Transactions on Circuits and Systems* 1988; **35**:1257–1272.
8. Espejo S, Rodriguez-Vazquez A, Dominguez-Castro R, Huertas JL. Smart-pixel cellular neural networks in analog current-mode CMOS technology. *IEEE Journal of Solid State Circuits* 1994; **29**:895–904.

9. Kinget P, Steyaert MSJ. A programmable analog cellular neural network CMOS chip for high speed image processing. *IEEE Journal of Solid-State Circuits* 1995; **30**:235–243.
10. Rodriguez-Vazquez A, Espejo S, Dominguez-Castro R, Carmona R, Roca E. Mixed-signal CNN array chips for image processing. *SPIE* 1996; **2950**:218–229.
11. Carmona R, Espejo S, Dominguez-Castro R, Rodriguez-Vazquez A, Roska T, Kosek T, Chua LO. A 0.5 μm CMOS CNN analog random access memory chip for massively image processing. *Proceedings of the Fifth IEEE International Workshop on CNN and their Applications*. London, U.K., April 1998; 243–248.
12. Etienne-Cummings R, Kevork Kalayjian Z, Cai D. A programmable focalplane MID image processor chip. *IEEE Journal of Solid-State Circuits* 2001; **36**:64–73.
13. Devos F, Zhang M, Ni Y, Pône J.-F. Treaming smart CMOS imager with tunnel effect non volatile analog memory. *Electronic Letters* 1993; **9**(20):1766–1767.
14. Eichenberger C, Guggenbuhl W. On charge injection in analog MOS switches and dummy switch compensation techniques. *IEEE Transactions on Circuits and Systems* 1990; **37**:256–264.
15. Canny JF. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 1986; PAMI-**8**:679–698.
16. Dupret A, Klein J.-O, Nshare A. A programmable vision chip for CNN based algorithms. *Proceedings of the Sixth International Workshop on Cellular Neural Networks and their Application*. May 23–25, 2000; 207–212.

