A vision system on chip for industrial control
From light to smart vision

Eric Senn and Eric Martin
L.E.S.T.E.R., University of South Brittany, BP 92116, 56321 Lorient Cedex, France

Abstract: This paper describes the building of an original vision system on chip. Our smart sensor performs motion detection and pattern recognition with only one single line of pixels. The maximum cross-correlation number is processed by the processors network. A peculiar methodology was defined to tackle the sensor's processing part design. Relying on our architectural synthesis tool GAUT, it leads to an optimal matching between the application specifications, the sensor's architecture and the processor's. The elementary processor's architecture is detailed. Its CMOS VLSI implementation is sketched, as well as the sensor's analog part and the light to byte conversion. The circuit's final structure and floorplan are outlined. Its performances are exhibited.

Key words: Vision System On Chip, Active Pixel Sensor, Industrial Control, Motion Detection, Pattern Recognition

1. INTRODUCTION

The least a vision system must include is one optical sensor and one processor. The sensor issues images to the processor that performs several low and high level operations to extract relevant information from the image flow. The amount of data to be transferred between those two components is huge. This transfer is thus very time and power consuming. To save some precious energy (as precious as the system ought to be mobile), it is necessary to reduce the bandwidth requirement between sensor and processor. This is achieved when some processing capabilities are given to the sensor, making it undertake every low-level task (*Figure 1*). As a second effect, the main processor load is alleviated; this processor can run slower, consumes less, and has plenty more time to take the high-level decision.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35597-9_40
M. Robert et al. (eds.), *SOC Design Methodologies*

Thus, Vision Systems On Chip (VSOC) actually permit to reduce the vision system's overall cost. The system is cheaper to build since a simpler interface between sensor and processor is needed, even the processor could be slow-down. It is cheaper to use because some energy is saved in the data transfer, or could be saved in the processor clocking.

A great number of VSOC, also called Active Pixel Sensors, or Smart Sensors, are made of a 2D array of pixels (a pixel is an elementary light sensor), connected to a 2D array of elementary processors, each processor being dedicated one or more pixels in the focal plane. Classically, every processor runs the same instruction on different data: the processors' network control is SIMD [4]. Such systems have a very bad filling ratio: the resolution has to be drastically decreased to make enough room for the processing part [3]. Indeed, the design of smart pixel sensors is strongly limited by the maximum transistor count and the available silicon area [8]. To include the SIMD array control inside the circuit is then not even thinkable. Our approach is quite different. We managed to keep VSOC' advantages by removing pixels from the chip. More silicon area is then available for more control and more processing capabilities. According to this idea, we designed smart sensors with only one single or two orthogonal lines of pixels, for motion detection and pattern recognition in the frame of industrial control.

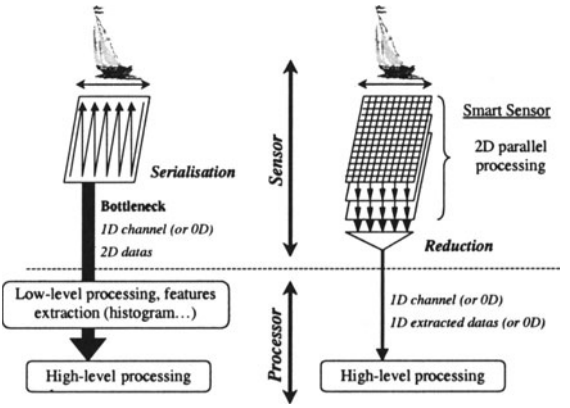

Figure 1. Vision systems: sensor plus processor (left), smart sensor (right)

In the following, we first explain how efficient image processing can be done with such sensors. Then we outline the methodology we defined to handle the smart sensor's processing part design. The resulting circuit's architecture is detailed afterward, and next the processor's. We finally sketch the sensor's VLSI implementation and give its foreseen performances.

2. PRINCIPLE

Assuming a few hypothesis depending on the application, it can be shown that the same amount of information is obtained with a full 2D array of pixels than by merging data from two orthogonal lines of pixels [1]. Consider *Figure 2-a*. The pixel's value pi=I(xi,yi) of image I is retrieved by merging the two projections xi and yi of pi at time t. An algorithm that handles a 2D data flow can thus be translated in an algorithm that operates on two 1D data flows, each of them being associated with the time variable t.

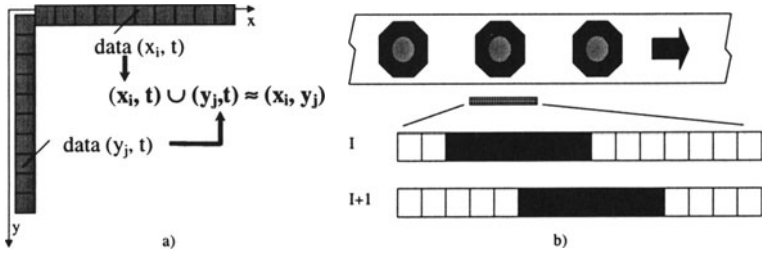

Figure 2. (a) 2D data flow equivalent to 2 x 1D data flows with time t - (b) Motion detection and pattern recognition for industrial control

This principle only holds on applications with bounded and known behaviour. If a "normal" motion can be determined, it becomes possible to detect a rupture in the motion or in the object shape [2]. In the frame of industrial control, motion detection and pattern recognition can even be done with a single line of pixels. On *Figure 2-b*, items of same shape and size are passing before the sensor at speed v (in pixels per second). The motion's direction is known in this case.

Our purpose is to measure this speed and to determine if an item has a fault in its shape. The two images I and I+1 on *Figure 2-b* are sampled at a time interval T. The cross-correlation number *Cxx* is calculated for every value of τ from 0 to N-1 (N is the number of pixels on the line).

The value of τ where *Cxx* is maximal gives the item's motion, which is easily related to its speed. Cxx_{Max}, the maximal value of Cxx, informs on the item's shape. Given a shape, Cxx_{Max} will indeed always be the same. Whenever Cxx_{Max} changes significantly, the item's shape is altered. The target's speed is $v = \tau / T$ (pixels per second):

$$C_{XX}(\tau) = \sum_{j=0}^{N-1} X_t(j) \cdot X_{t-T}(j-\tau)$$

3. METHODOLOGY

Methodologies for algorithm/architecture matching classically start with the application description, both an architectural model and an implementation method, then issue a solution that can be software, hardware, or mixed. Never would they tackle sensors' implementation. Acquisition capabilities are however of a great use in many systems. The methodology we proposed copes with systems that include sensors, like VSOC. Such a system can be decomposed in three parts: algorithm, processing, and sensor (*Figure 3-a*).

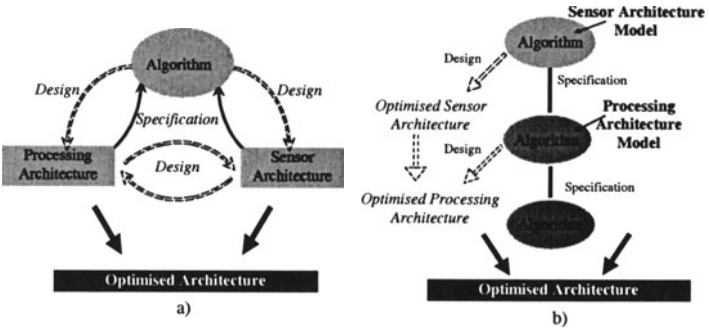

Figure 3. (a) Algorithm/Architecture matching (b) Design flow

Each part has a bi-directional "design/specification" relation with any other. To build a linear design flow involves to break these relations as shown in *Figure 3-b*. The flow is re-iterated until resulting architectures meet the specifications. The final design-flow is shown on *Figure 4*.

As stated on this figure, every architectural model entails a new formulation for the algorithm. Once its correctness checked, this algorithm is used as an entry to GAUT.

GAUT is an architecture synthesis tool for dedicated signal processors [6]. It issues a RTL netlist from an algorithm and an associated timing constraint (area constraint can be specified too). The timing constraint we choose was to obtain a processor as fast as the light to byte conversion. The amount of treatment for one processor to achieve depends on the number of pixels P it copes with. GAUT firstly selects operators depending on the timing constraint. The necessary hardware resources and the involved silicium area are computed. The optimal number of pixels P is determined in this very first step [2]. P is set to minimize the processing area per pixel Sp. With St the total processing area and Spe the processor's area, one gets:

$$Sp = \frac{St}{N} = \frac{Spe}{P}\text{; indeed: } St = Spe \cdot \frac{N}{P}$$

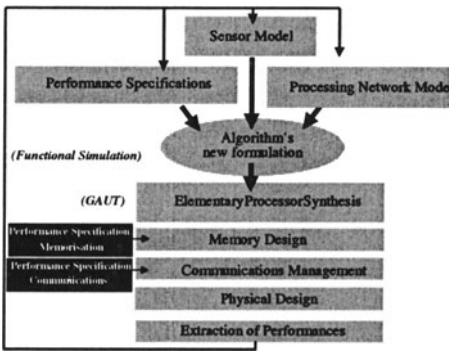

Figure 4. Detailed design flow

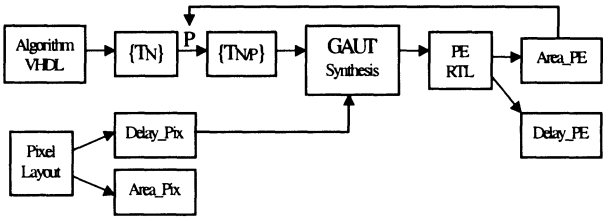

Figure 5. Synthesis tool GAUT and the methodology

The processor's RTL netlist is obtained afterward and processed by a commercial logic synthesizer to target VLSI. *Figure 5* sketches the methodology. The algorithm is written in VHDL. The set of treatments {TN} represents the work to be done in the algorithm, for all pixels. The subset of treatments {TN/P} represents the elementary processor's work and depends on the parameter P. It is synthesised by GAUT with the pixel's delay as timing constraint. The resulting processor's area is computed. The processus is repeated with different values for P until the minimum PE area is found. The computed PE delay matches the timing constraint.

4. CIRCUIT'S ARCHITECTURE

The correlation algorithm is strongly regular and can be easily distributed between every processor on a SPMD network :

Iterate
For "all new image"
Sequential
For $\tau = [\tau_{min} ; \tau_{max} ; \tau+1]$
***PARALLEL* for all k = 1, ...N/p**
$C_{xx}^{k}(\tau) = 0$
For j = [(k-1)p ; kp ; j+1]
$C_{xx}^{k}(\tau) = C_{xx}^{k}(\tau) + X_{I1}(j).\ X_{I0}(j-\tau)$
End j ; End k
***SEQUENTIAL* for k = 1, ...N/p**
$C_{xx}(\tau) = 0$
$C_{xx}(\tau) = C_{xx}(\tau) + C_{xx}^{k}(\tau)$
End k
End τ
Max $[C_{xx}(\tau)]$ => ***Motion*** $= \tau$

The cross correlation number Cxx(τ) is calculated for every value of τ (from 0 to a parameter τ_{Max}). N is the number of pixels on the line, P the number of pixels by elementary processors in the circuit. $X_{I1}(j)$ is pixel j from image I1. Image I0 follows I1. During the parallel part, each processor computes a part of the cross-correlation number for N/P pixels. For instance, processor k tackles pixels (k-1)p to kp. The last sequential part is necessary to add up all the P parts of the cross-correlation number and to finally compute its maximal value among all the τ values from τ min to τ max.

Our purpose here is to determine the optimal number of pixels per elementary processor, in other words, to match the algorithm with an architecture, considering two constraints: silicon area and circuit's speed.

According to the methodology, several high level synthesis were performed using GAUT with different values for P, and the pixel cell's speed as timing constraint (see the VLSI implementation section). The minimum processing area per pixel was obtained with P=8 [9]. 4 processors are thus necessary to build our 32 pixels prototype. The circuit's architecture is outlined on *Figure 6.*

Every processor stores two successive images (I0 and I1) and computes the partial sum $C_{XX}^{k}(\tau)$ from these images and its right neighbour's partial sum. Result is send to the processor's left neighbour. The last processor of the line finally issues the cross-correlation coefficient for the chosen τ value. Note that Cxx is computed from right to left in the processors line. The same result would have been obtained from left to right. The τ value where Cxx(τ) is maximal gives the item's motion. The MAX module keeps the maximal value for Cxx: Cxx_{Max} .

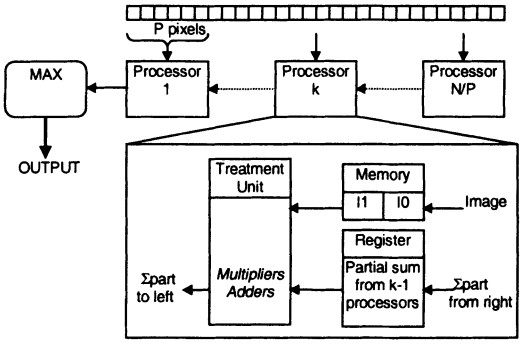

Figure 6. The circuit's architecture

5. PROCESSOR'S ARCHITECTURE

As shown on *Figure 7*, the processor contains sixteen, 8 bits registers (parallel and serial load), two 8 bits multipliers, and one 16 bits adder.

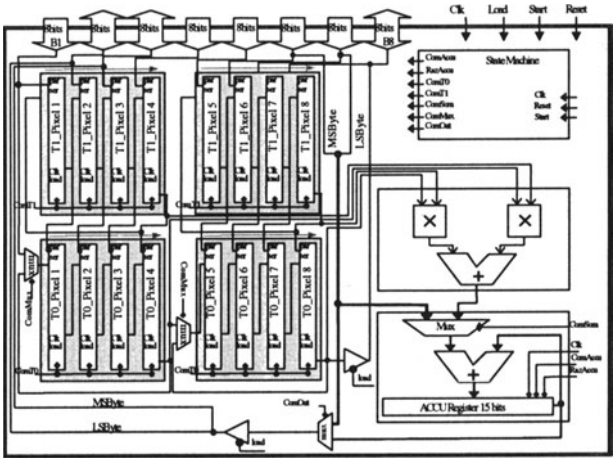

Figure 7. The processor's architecture

Pair of registers are connected vertically to make eight 2 stages FIFO, which store the pixels value for images I0 and I1. I0 is the oldest image and is stored in the 8 lowest registers called T0-pixel1 to T0-pixel8. I1 is the last sampled image. It is stored in the 8 upper registers on the figure, called T1-pixel1 to T1-pixel8. T1 is the first stage of the FIFO, T0 the second stage.

Horizontally, registers are connected by four in a ring in which data can shift to reach the processing unit. This minimises the multiplexers count.

Eight data busses reach the elementary processor. Their main purpose is to input the pixels' value from the light sensor and the analog to digital converter. However, most of them are multi-purpose. B1 also inputs the register T0-pixel1 value from the processor's left neighbour. B2 and B3 issues the processor's partial sum to the left neighbour. B6 and B7 get this partial sum from the right and B8 issues the right register T0-pixel8 value to the right neighbour.

The circuit's behaviour is summarised below:

Step1, *Initialisation*: Sample image. Store image to registers T1.

Step2, *Shift and store*: Sample image. Shift T1 to T0 (vertical FIFO). Store image to registers T1. At this stage I0 is in T0, and I1 in T1.

Step3, *Cross correlation calculation*: Read pixels' values in T0 and T1, (τ=0). Horizontally shift pixels value to reach multipliers (internal four registers rings are used). Resulting partial sum is accumulated in the ACCU register.

Propagate partial sum across the line. Issue Cxx(τ) .

Step4, *Increment τ value*: Horizontally shift to the right every registers T0 in all processors. Internally shift-right registers T0. Send T0_pixel8 register's value to the right. Receive T0_pixel1's value from the left neighbor.

Step5: If $\tau <= \tau_{Max}$ go to step3.

Step6, *Compute* Cxx_{Max} : Issue Cxx_{Max} and the corresponding τ value.

Step7, *next image*: Go to step2.

6. VLSI DESIGN

A great care was taken in designing the sensor's analog part. This part is absolutely essential for the first thing a vision sensor has to do is to convert light to byte. This conversion is not so easy to achieve for light has a very broad dynamic, and because the sensor could be used in many different places, with different light average and max/min levels. The basic light sensor we used is a 60 x 60μm photo diode. It provides an output current iL ranging from 10^{-11}A to 10^{-6}A for a light level from 0.01 (dark) to 100 W/m² (broad daylight) [10]. *Figure 8-a* shows the pixel cell's structure.

Transistor M0 converts current iL in voltage v1. Wired like a diode, it behaves in the subthreshold region and has a logarithmic response [11]. Transistor M1 acts as an amplifier. First, Exp and Preb are low so that Vout=Vdd. Then Preb is high and Cout is decharged across M2 while Exp is high. Vout final value is linear with ln(iL) (see *Figure 9*). When Exp is

toggled high, charge sharing induces a parasitic current from Exp, via M2 and M1, to M1's gate. For very low iL, this current increases v2 during exposure time, and finally leads to bad output values. To keep the sensor's response linear, M2's length was increased to 4μm.

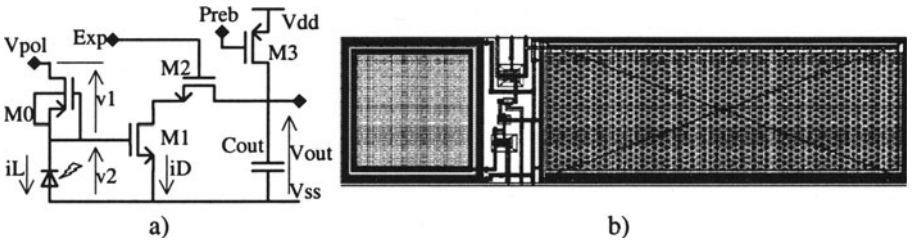

Figure 8. (a) The pixel cell's structure: photo-diode and response "linearization" - (b) The pixel's layout

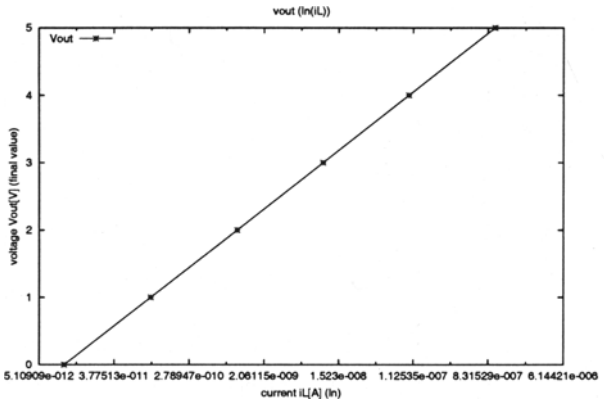

Figure 9. Simulated Vout(ln(iL))

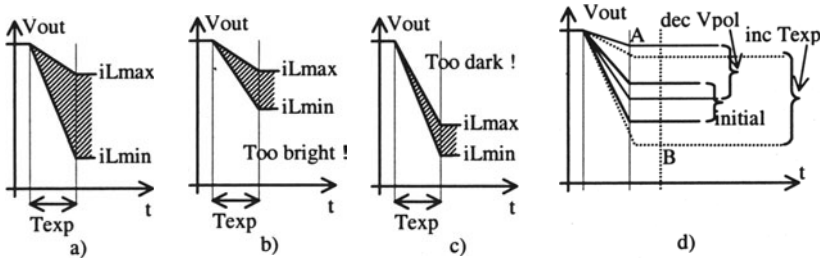

Figure 10. Ouput dynamic for (a) normal (b) over exposed (c) under exposed scene - (d) Trimming procedure for Vpol and Texp

Exp is high during time Texp. Texp is somewhat like a camera's exposure time. Along with Vpol, it allows for trimming the sensor's output dynamic. Indeed, transistor M1 converts v2 to iD. Since v2 equals Vpol minus v1, iD's output range actually depends on Vpol.

As shown on *Figure 10*, the sensor's output dynamic is maximal only if the image has good contrast. It is poor if the scene is too bright or too dark. In the first case, output dynamic can be restored merely by increasing Texp. It is restored by decreasing Vpol in the second case. Both of these actions can be used, as shown on *Figure 10-d* where the initial output dynamic is too narrow. Hence the trimming procedure : first to decrease Vpol down to A, then to increase Texp up to B.

The final pixel's layout is given on *Figure 8-b* (65.1 x 256.8μm). The photo receptor is the square on the left (64.6 x 64.6μm). Cout is the large rectangle on the right (65.1 x 166.8μm). It was set to 15pF to overcome the cumulated input capacitance of the multiplexing part to the ADC. Indeed, Cout is set to reduce charge sharing during multiplexing [9]. It must not be too big to avoid wasting room on the floorplan, and to keep the sensor's response time small.

The processor's layout can be found in [9]. It is 1.1mm wide, its surface area is 1.25mm^2. This processor performs one cross-correlation in 200ns. The AMS CYX 0.8μm process (2 polysilicium, 2 metal layers) was used.

The floorplan is constrained by the pixels' location on chip. Pixels have to be placed in a line. The analog to digital converter (ADC) we used, the ADC02 (612 x 358μm), comes from the AMS library and is almost as big as the processor itself. To avoid wasting place, we multiplexed one single ADC to all pixels.

To bring sampled pixels' values to the processors, a pipelined bus was designed, as shown on *Figure 11-a*. This bus behaves somehow like a FIFO. The first sampled data has to cross all the stages to reach the last processor. The operational amplifier (OP5B's input capacitance is 0.1pF) is needed to hide the converter's huge input capacitance (8pF) to the transmission gates which supports multiplexing. Extra input and output are added for test purpose. It is so possible to read or write the pixels' pipelined bus from outside the circuit. The pixel-max and pixel-min modules store the maximal and minimal pixels' values in an image. This values are used to bias the sensor's output dynamic afterward. If the max value is to low, Vpol should be decreased. For the analog to digital converter's output is 8 bits wide, a too small max value could be when pixel-max<250d. Also, if the min value is to high, Texp should be increased. For the same reason, a too high min value could be pixel-min>05d.

Final performances actually depend on the multiplexing ratio. The AMS ADC02 performs one conversion in 50μs; it will do the whole line in

1600μs. The chip's sampling frequency is then Fs=625 images per second. It is not bounded by the processor's speed: one cross-correlation is issued every 200ns and the max across 32 τ values is found in 6.4μs. The chip's performances are neither settled by the processor's speed, nor by the pixel cell's acquisition time (Texp is about 0.5 μs). They are mainly settled by the ADC's response time and the multiplexing ratio. The sampling frequency could be set to 2500 images per second if four ADC were used instead of one. No multiplexing is needed in that case.

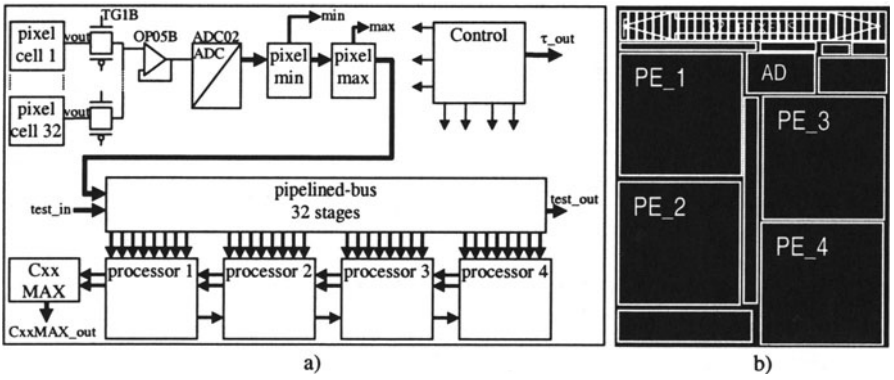

Figure 11. (a) The circuit's synopsis (control lines are not represented) - (b) The floorplan shows the sizes of processor (PE), analog to digital converter (ADC) and pixels (Pix)

The maximum target's speed is related to the sampling frequency. If the target's motion is more than 32 pixels in 1600μs, it will not appear on the next sampled image. Hence, the maximum speed is $32/1600^{E-6}$ = 20000pixels/second. The circuit's floorplan is sketched on *Figure 11-b*. The core is about 3mm x 2.4mm.

7. CONCLUSION

This paper shows that motion detection and pattern recognition are possible with only one single line of pixels, assuming a few hypothesis, as in the presented frame of industrial control. Moreover, by removing pixels, enough room is made on chip for the whole processing and control resources. As a result, dedicated complex algorithms can be implemented though the sensor remains fully independent. Our methodological approach was very useful in designing the vision system and could be re-used in many smart sensors' design. Given timing constraints, the optimal number of processors per pixel was found. The sensor's light to byte conversion

exhibits original mechanisms to balance the analog part's output dynamic, depending on the input average light level. The pixel's response time, 0.5μs, was used as a timing constraint to the processor's architecture synthesis. This synthesis was performed with our architectural synthesis tool GAUT. The processor's RTL netlist is implemented in CMOS VLSI thanks to commercial logic synthesis tools. The processing time for 32 pixels is 6.4μs. To reduce the circuit's surface area, one single analog to digital converter was used and multiplexed between all pixels. As a result, the circuit's maximal sampling frequency is decreased to 625 images per second.

8. REFERENCES

[1] Emzivat D., Gagnadre C. and Martin E. "Vision sensor for the industrial quality control". *Proceedings of the 7th International Conference on Image Processing and its Application*, Manchester, UK, 12-15 July 1999.

[2] Emzivat D., Gagnadre C. and Martin E. "Optical integrated circuit for the quality control". *International Symposium on Microelectronic Manufacturing Technologies*, Edimbourg, 19-21 May 1999.

[3] Gruev V. and Etienne-Cummings R.R. "A programmable spatiotemporal image processor chip". *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, May 28-31 2000.

[4] Ishikawa M., Komuro, T., Ogawa K. and Ishii I. "A CMOS vision chip with SIMD processing element array for 1ms image processing". *Abst. 1999 Dig. Tech. Papers of 1999 IEEE Int. Solid-State Circuits Conf.*, pages 206-207.

[5] Komuro T., Ishii I., Ishikawa M. and Yoshida A. "High speed target tracking vision chip". *Proceedings of the Fifth IEEE Int. Workshop on Computer Architecture for Machine Perception*, Padova, Italy, September 11-13 2000, pages 49-56.

[6] Martin E., Sentieys O., Dubois H. and Philippe J.L. "GAUT, An architectural synthesis tool for dedicated signal processors". *Proceedings of the EURO-DAC*93, 1993.

[7] Paillet F., Mercier D., Bernard T.M. and Senn E. "Low power issues in a digital programmable artificial retina". *IEEE Alessandro Volta Memorial International Workshop on Low Power Design*, Como, Italy, March 4-5, 1999.

[8] Paillet F. and Mercier, D. "Design solutions and techniques for vision system on a chip and fine-grain parallelism circuit integration". *Proceedings of the Thirteenth Annual IEEE International ASIC/SOC Conference*, Arlington VA, USA, September 13-16 2000.

[9] Senn E., Emzivat D. and Martin E. "A smart pixel sensor for industrial control". *International Symposium on Signals, Circuits and Systems*, Iasi, Romania, July 10-11 2001.

[10] Sicard G. "From biology to silicium: A bio-inspired analog retina for smart vision sensor". *PhD Thesis*, Institut National Polytechnique de Grenoble, 1999.

[11] Uyemura J.P. "Circuit design for CMOS VLSI". Kluwer Academic Publishers, 1992.