A 3-D Chip Architecture for Optical Sensing and Concurrent Processing

Angel Rodríguez-Vázquez [1,2], Ricardo Carmona [1], Carlos Domínguez Matas [2], Manuel Suárez-Cambre [3], Victor Brea [3], Francisco Pozas [1], Gustavo Liñán [1], Peter. Foldessy [4], Akos. Zarandy [4] and Csaba Rekeczky [5]

[1] IMSE-CNM/CSIC and Universidad de Sevilla (SPAIN)
[2] AnaFocus (Innovaciones Microelectrónicas S.L.), Sevilla (SPAIN)
[3] Universidad de Santiago de Compostela (SPAIN)
[4] MTA-SZTAKI (HUNGARY)
[5] Eutecus, Inc, Berkeley, CA, (U.S.A.)
angel@imse-cnm.csic.es
angel.rodriguez-vazquez@anafocus.com

ABSTRACT

This paper presents an architecture for the implementation of vision chips in 3-D integration technologies. This architecture employs the multi-functional pixel concept to achieve full parallel processing of the information and hence high processing speed. The top layer includes an array of optical sensors which are parallel-connected to the second layer, consisting of an array of mixed-signal read-out and pre-processing cells. Multiplexing is employed so that each mixed-signal cell handles several optical sensors. The two remaining layer are respectively a memory (used to store different multi-scale images obtained at the mixed-signal layer) and an array of digital processors. A prototype of this architecture has been implemented in a FDSOI CMOS-3D technology with Through-Silicon-Vias of 5μm x 5μm pitch.

Keywords: 3-D Optical Sensors, Vision Systems, Navigation Applications

1 INTRODUCTION

Vision systems are among the most challenging of the application drives mentioned by ENIAC´s SRA [1] and the International Technology Roadmap for Semiconductors (ITRS) [2]. The design of imaging systems (sensors + readout + data conversion + controller + drivers) on CMOS chips has been making good progress during the last decade [3]. The main design target for CMOS imaging chips is reproducing images with given accuracy and speed. The target for vision systems is different. Similar to imagers, they have 2-D light intensity maps as inputs. Also, they may output images for monitoring purposes. However, their primary outputs are not images, but reaction commands. For instance, these commands may be needed to discard defective parts following visual inspection in a production line; or to trigger evasive maneuver following the visual detection of looming objects moving into collision course towards a vehicle; or to align unmanned aerial vehicles while landing in a platform following the signaling provided by a set of light beacons; or to trigger alert mechanisms if suspicious events are detected into a scene subjected to video surveillance; just to mention some examples.

Vision applications require to complete the full "sense → process → analyze → make decision" cycle. It involves *large amount of data*, especially in applications where high frame-rate is essential. Making a real-time decision also requires *low latency* from the system, which renders the analysis of the large input data set even more demanding.

Optical Sensing and Detection, edited by Francis Berghmans, Anna Grazia Mignani, Chris A. van Hoof,
Proc. of SPIE Vol. 7726, 772613 · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.855027

The industrial state-of-the-art considers vision systems as "seeing computers" or "computers that see". This manifests itself at the architecture typically used for them, namely: an *imager* (image sensor) to acquire and digitize the sensory data and a *host processor* to handle this huge amount of raw data. Such brute-force approach does completely ignore the specifics of the data, the ways how interesting pieces of information emerge from the data, and hence results in largely inefficient systems.
Not only conventional computer architectures are inadequate. Conventional algorithmic solutions used in these architectures are also inadequate. This fact has been highlighted in a very recent paper published in *Vision System Design* [4]. It states that brute force pattern matching, the conventional approach adopted by many system developers, is not the right tool in many applications. Instead, sic, "*a majority of smart camera applications can be solved using only a small number of image processing algorithms that can be learned quickly and used very effectively*" [4]. Interestingly enough these simple algorithms (thresholds, blob analysis, edge detection, average intensity, binary operators, …) can be mapped down onto dedicated, processor architectures composed of simple processors with mostly local interactions – the sort of architectures addressed by this paper.

During the last few years authors have worked on mapping these simple algorithms (thresholds, blob analysis, edge detection, average intensity, binary operators, etc.) onto dedicated processor architectures composed of simple processors with mostly local interactions. Programmable sensor-processor chips have been devised and implemented using conventional CMOS technologies [5] [6] [7] [8]. These chips consist of 2-D arrays of *multi-functional pixels* which perform full parallel-processing of the incoming image flow to allow very high operation speed. Using these chips at the front-end, complete, autonomous Vision-System-on-Chips (VSoC) have been realized which are capable of going from sensing to decision-making at thousands frames-per-second speed [8].

A key feature of these architectures is the use of *multi-functional pixels*; i.e., pixels that embed functional structures other than those used for sensing (memories, arithmetic, thresholding, etc.). Because the sensing area of these pixels is a small fraction of the total pixel area, the *fill factor* decreases thus impacting on the spatial resolution and the optical sensitivity. This drawback can be precluded by resorting to the use of 3D integration technologies and splitting the multi-functional feature of the pixels among several layers: for sensors, for analog read-out and mixed-signal pre-processing; for memory and for digital processing. This paper presents a generic 3D architecture for vision.

2 CONCEPTUAL 3-D SYSTEM ARCHITECTURE

The drawing at the top in Figure 1 shows a conceptual representation of the type of architectures addressed in this paper. Basic ideas underlying this architecture are:

- The hierarchical decomposition of the processing chain, with different layers handling different kind of data.
- The use of distributed, parallel-processing at the different layers and particularly at the front-end stages.

These ideas fit very well to the nature of the vision processing chain [8], illustrated at the bottom in Figure 1. The figure includes several processing steps and shows that the amount of data decreases as information travels along the chain, namely:

- At initial steps the number of data is huge and many of the data are redundant and hence useless to the purposes of reaction prompting.
- As information flows across the processing chain and abstract features are extracted from the incoming images, the number of data decreases.

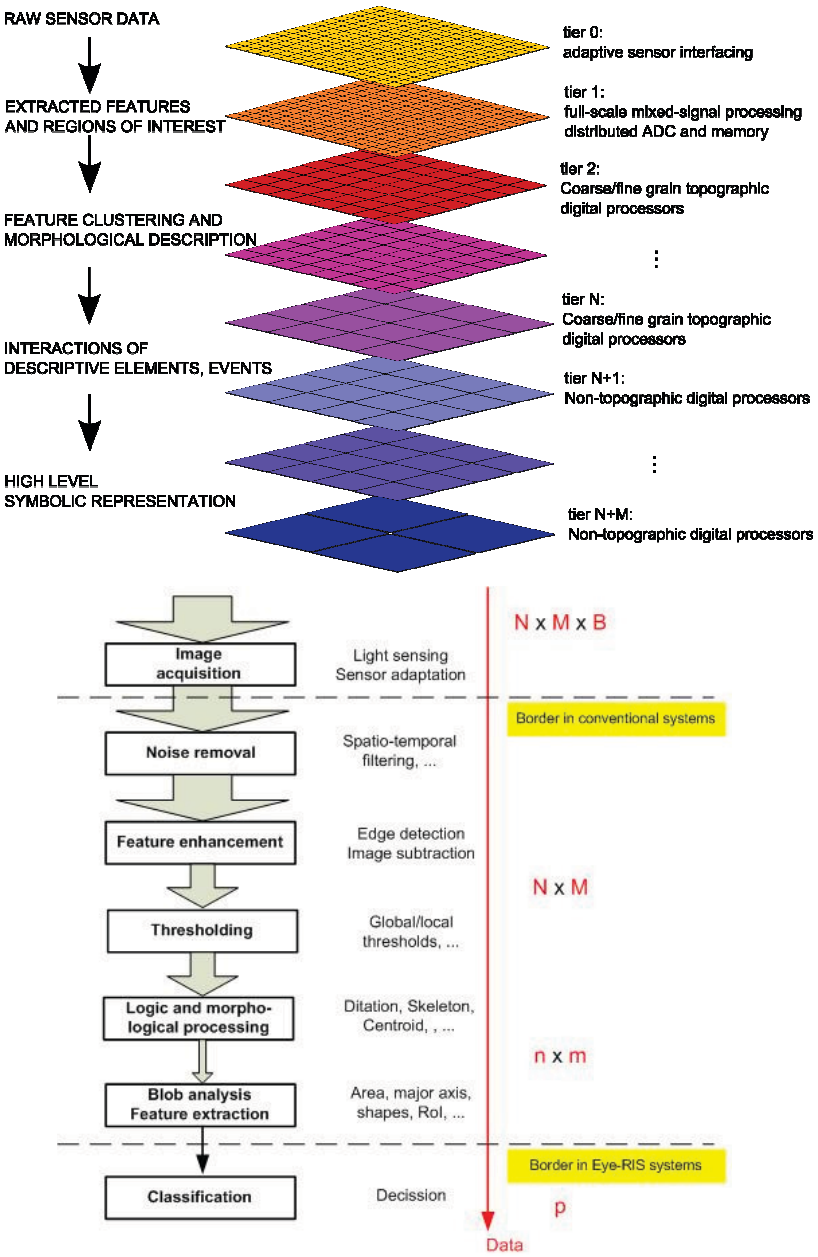

Figure 1. Generic 3-D Architecture and processing chain of vision. As data evolve from the sensor interface (raw data), the amount of data decreases and the abstraction level increases. N represents the number of columns, M the number of rows and B the number of bits used per pixel data: n<N; m<M and p<(n,m).

3 THE VISCUBE CHIP ARCHITECTURE

The so-called VISCUBE chip architecture is a particular case of the general architecture in Figure 1. Figure 2 shows two views of of this specific architecture.

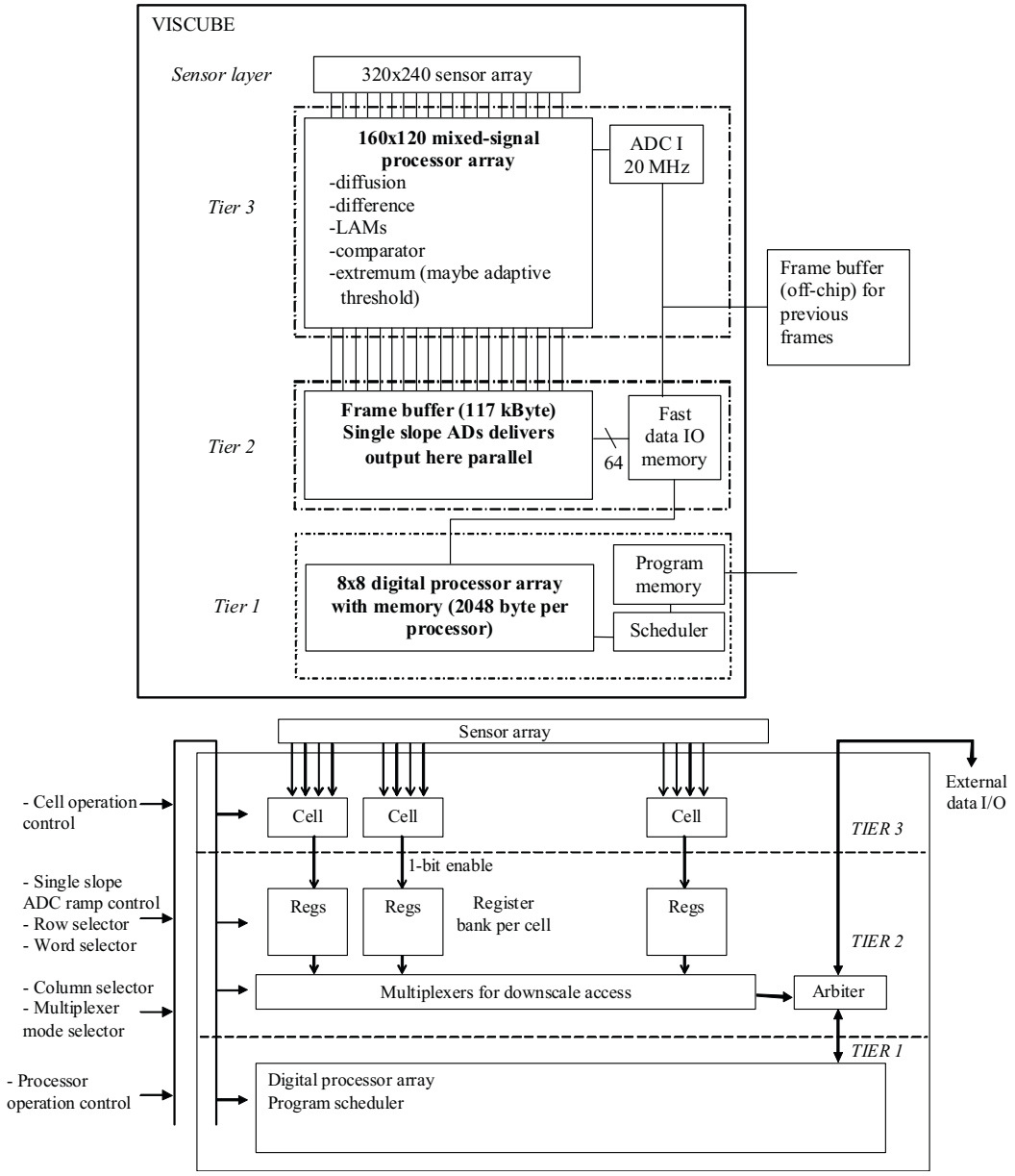

Figure 2. VISCUBE Chip Architecture

Figure 2 shows that the architecture consists of four layers. Namely: one sensor layer which is bump bounded to a three-tier structure composed of a mixed-signal layer (Tier 3), a memory buffer (Tier 2) and a digital processing layer. The last

three layers are interconnected via Through-Silicon-Vias with 5µm x 5µm pitch. The architecture employs fine-grain topographic processors in the mixed-signal layer [8] and coarse-grain topographic processors in the digital layer [10].

The sensor is implemented on an extra semiconductor layer, which is connected to the mixed-signal layer via bump bonding. The sensor resolution is 320 x 240. The sensor material will be most probably InGaAs, however silicon is also considered. In the former case, the imager will be sensitive in the SWIR range, while in the latter case in the visual range. The sensor layer contains the photodiodes only. All the other circuitry (amplifier, switches, reset circuit, etc.) will be implemented on the mixed-signal layer.

The sensor pitch is 25micron. The most important technical parameters of the InGaAs sensor will be:

- Wavelength: 0.9 - 1.7 mic
- QE: ~80% (+-10%)
- Dark current will be ~2nA/cm^2 for the detector diode at 20C ~ room temperature, Rev. bias: -0.1V; (~3nA/cm^2 for -0.4V)
- ~2nA/cm2 dark current density corresponds to ~ 50e read-out noise
- Expected dynamic range: >4 orders of magnitude ~14 bits (meaning saturation at roughly ~ 500 000e)
- Diode capacitance: <35 fF (for 25 micron pitch)

4 ALGORITHMIC PROCESSING IN THE VISCUBE CHIP

The primary application of the VISCUBE is airborne visual navigation and reconnaissance. These applications are based on segmentation. However, segmentation of an image on a moving platform requires image registration too. In this context, image registration means to find and calculate the affine transformation compensating the ego motion of the camera. To that purpose different scales must be employed and he algorithm steps are performed in different image scales (resolution) distributed among the analog processor array, the digital processor array, and in the external system.:

- Scale 0: Sensor resolution, 320 x 240 grayscale, 8-bit representation
- Scale 1: Mixed-signal processor resolution, 160 x 120 grayscale, 8-bit
  Mixed-signal processor resolution, 160 x 120 binary, 1-bit
- Scale 2: Downscaled Scale 1 resolution, 80 x 60 grayscale, 8-bit

The algorithm does four basic steps:

- Feature selection (mixed-signal layer)
- Displacement calculation (digital signal layer)
- Feature extraction (digital signal layer)
- Registration of consequent frames, tracking of moving objects (external processor).

The first three steps are supported by the Viscube chip.

Downscaling is achieved through two independent actions namely apply a low-pass filter and generate smaller image by skipping every second (or more) rows and columns. The low-pass filtering is done in the mixed-signal processor layer. The low-pass filtering has two simple types:

- Scale 0 to Scale 1 transition by averaging in analog or digital domain by binning the four integration capacitors or alternatively using a binary adder in the frame-buffer.
- Scale 1 to Scale 2 uses the resistive grid.

Further low-pass filtering can be done by reinitializing the resistive grid. The image resolution decrease is achieved at readout time from the framebuffer by applying differently sub-sampling addressing types

The algorithmic steps are:

1. Image acquisition at Scale 0, 320 x 240 resolution.
2. Conversion of Scale 0 analog representation to digital, stored in the framebuffer.
3. Generation of Scale 1 image by averaging four pixels of the Scale 0 image stored in a mixed-signal processor.
4. Start diffusion process, during the diffusion process a snapshot is stored in a temporary location, of which is subtracted from the final diffusion result.
5. The local maximum and/or minimum positions are detected on the difference. The result is a binary Scale 1 image, which is stored in the framebuffer as 1-bit image.
6. Conversion of the diffusion result at Scale 1 resolution and stored in the framebuffer (that is Scale 2 content).
7. The content of the framebuffer is accessed by the digital processor array or the external system.
8. Scales are read out as complete images
9. Scales are read out in small windows at random locations.
10. Possible repetition of Step 4-7.

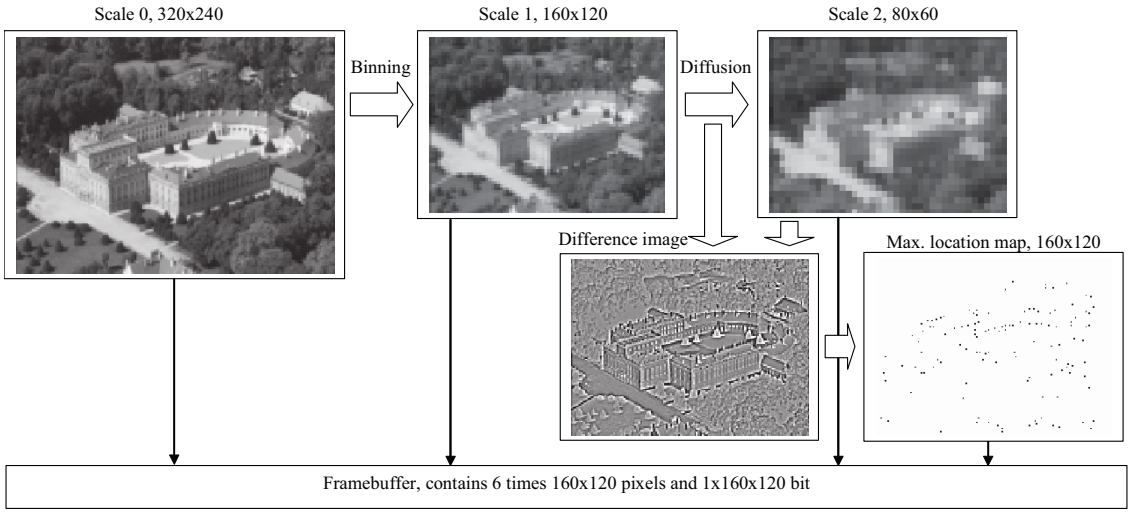

Figure 3. Sample images of the algorithm executed on the mixed-signal processor layer (simulation). The scale 2 read out from the framebuffer is an 80 x 60 sized image.

5 TIER-3: MIXED-SIGNAL PROCESSOR LAYER

Each cell in the mixed-signal layer (MSPE) provides access points for 4 diodes/sensors – see Figure 4. The MSPE architecture includes a transimpedance amplifier to adapt the sensor signal, Local Analog Memories (LAM´s) to store

the intermediate data, a programmable difussion network to perform Gaussian filtering, a susbtractor to obtain the Gaussian pyramid, a local minimum and maximum locator and a single-ramp Analog Digital Converter.

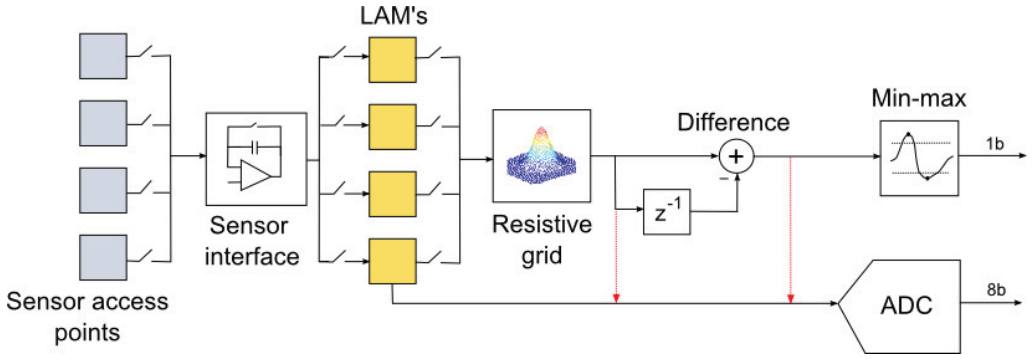

Figure 4. Cell architecture of the mixed signal processor array layer

Sensor Interface

The sensor interface is multiplexed in time, so only one quarter of the complete frame (320 x 240) will be captured at exactly the same time instant. This sensor interface has two stages, namely: a front-end composed of a capacitive transimpedance amplifier (CTIA) with offset sampling, and a SC circuit (LAMs) where the voltage data are stored. LAMs are also used to realize some arithmetic operations involving the pixels data (average, subtractions …).

The Capacitive Transimpedance Amplifier (CTIA) architecture consist of a discrete time integrator (Figure 5). The CTIA integrates the I_{ph} during the Φ_{int} phase and provides the following output:

$$V_o = V_{init} + \frac{I_{ph} * T_{int}}{C_{int}}$$

which can be tuned by changing the integration time.

LAMs are designed by using a sample and hold structure (Figure 6), which contains some extra switches to perform arithmetic operations. Φ_{av01}, Φ_{av02} and Φ_{av23} switches are used to perform the average of the four sensor. Φ_{difl} and Φ_{difl2} are used to connect indenpendly the different capacitances with the diffusive network. Finally the switches Φ_{MT}, Φ_{MA0} Φ_{MA1}, Φ_{MA2}, Φ_{MB3} and former ones reconfigurate the capacitors to calculate the substraction of the stored data.

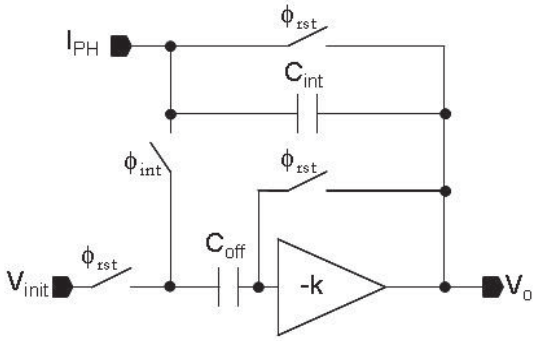

Figure 5. Transimpedance sensor interface

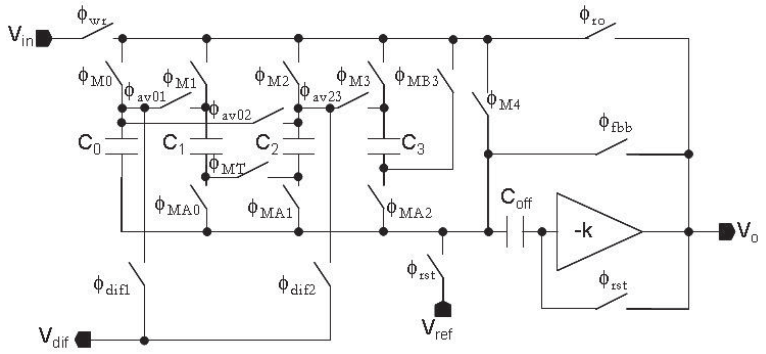

Figure 6. Discrete-time sample & hold amplifier

Diffusion network for Gaussian pyramid generator

Gaussian filtering of the quarter image (160 x 120) is realized by a Switched-Cpacitor (SC) circuit. This circuit allows the transference of charge packages among nearest neigbours, thus emulating the behavior of a Gaussian filtering.

The diffusion of the node voltages will be sampled at different point of the discrete evolution. These points will correspond to a particular scale representation. The accuracy of this correspondence will be given by the performance of the SC emulator. The samples of the diffusing image will be either processed to calculate significant points on the Gaussian pyramid, or combined to generate the Laplacian pyramid, or converted to be stored because of the needs of a particular algorithm. In this last case the diffusion should be stopped to avoid information losses. In any case, the user must be able to configure the way in which the intermediate stages of the diffusion are processed, combined or stored.

Pipelined Laplacian pyramid extraction

A possible combination of the diffusion intermediate results is to subtract each one from the following one. This is, to take the difference of two Gaussians (DoG), in order to obtain a differential scale space representation. This is also referred to as the Laplacian pyramid. A simple circuit to sample the evolution of the diffusion and obtain a serial stream of pixel values for successive Laplace representation scales is a couple of interleaved SC amplifiers. By avoiding the reset phase of the first capacitor, the output ids the difference of the input signal at two successive clock phases. By interleaving these blocks, we can obtain the successive scale values as a continuous stream.

By properly activating the clocks in these blocks, a different combination of representations at different scales could be computed.

Local extreme values detector

At any moment in the algorithm, each stored voltage can be compared with the equivalent voltage at a neighboring node and/or with a global threshold. In order to avoid the implementation of 8 comparisons at each pixel, we chose to time-encode the pixel value, making use of the available comparator at each MSPE. The combination of these comparisons with the appropriate logic will determine which pixels are local maxima or minima. Figure 7(a) and Figure 7(b) show the block diagram that are used to calculate local maximum and minimum respectively.

Therefore, the additional circuitry to incorporate to the MSPE is the 8-input NOR, and the inverter for the local minimum case, the two 2-input NOR gates, and the delay element, that can be a latch, making use of the clock that we already have at the cell. Then several switches are employed to reconfigure the circuit between the maximum detector and the minimum detector operation modes. And also, some switches are needed to reconfigure the output of the cell to deliver the min/max flag and not the conversion pulses, and to send the appropriate comparator output to the neighbors. An

additional AND gate is employed to force a transition to a low output when the 'end of the ramp' control signal arrives to the MSPE.

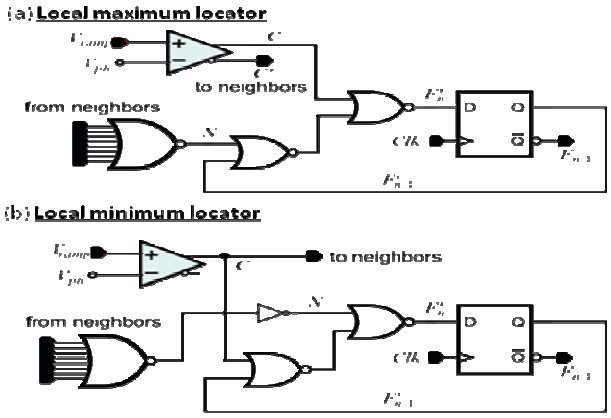

Figure 7. (a) Local Maximum Locator (b) Local Minimum Locator

ADC and inter-tier connections

The ADC will be a per cell single slope converter, which will be implemented partially in the 3rd and in the 2nd tiers. The comparator of the converter will be in the 3rd tier, while the latch will be implemented on the second tier (Figure 8). Note that only one wire is needed between the two main components of the ADC (one per ADC). This per cell interconnection between Tier 3 and Tier 2 will be implemented with a dense via array.

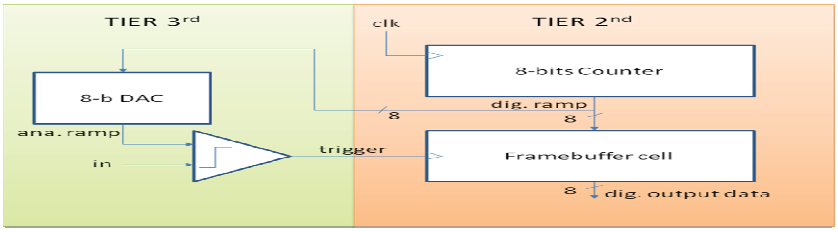

Figure 8. Single slope AD. The comparator (left hand side) will be implemented on the 3rd tier in every cell, while the8 bits memories (6 for each cell) will be located in Tier 2.

The most relevant design issue for the ADC concerns the comparator. Different architectures have been researched looking for compact implementations. The topology of selected is a compact and low-power solution and its power conssumption (Figure 9).

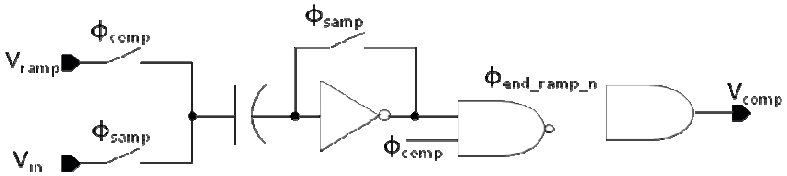

Figure 9. Comparator block diagram

6 TIER-2: FRAME BUFFER LAYER

The frame buffer architecture is defined as connection to the mixed-signal layer, the way of access to it from the processor layer, the bandwidth requirement, and the storage capacity.

Characteristic features

- Mixed-signal processor and framebuffer registers are in topographic connection (pitch-matched).
- The framebuffer get parallel write enable from the mixed-signal processor layer.
- The framebuffer can store 6x8-bit values and 2x1-bit value for each mixed-signal processor to store one-one Scale 0, 1, 2 grayscale and Scale 1 binary images.
- 114 Kbytes in total (117600 bits)
- The output of the framebuffer is 32-bit bus.
- The readout of all scales from the framebuffer is random access.
- The 32-bit word organization – how nearby 8-bit registers are grouped – is custom (see next chapters).
- 300-400 Mbytes/sec output rate. 32-bit bus of 80-100 MHz.
- The framebuffer stores data for any time
- Static or self-refreshed architecture.

Frame buffer architecture

The architecture is defined taking into account the storage needs and the scale readout schemes. The output (access) organization is somewhat special due to the downscaled access. Asides from the whole original and downscaled images windows at random positions are needed as well. These windows of size (24 x 24 or 32 x 32 pixels) can be anywhere in the frame. Hence, the addressing must be random access (no pointer, counter included).

Figure 10 shows the register bank. This will be a pitch matched design with the mixed-signal cells. There is one register bank below each mixed-signal processor cell, receiving the trigger signal from the mixed signal layer (Figure 11). The size of the register bank array will be 160*120. Each bank contains 6 bytes and 2 bits. This is about 117kByte of memory.

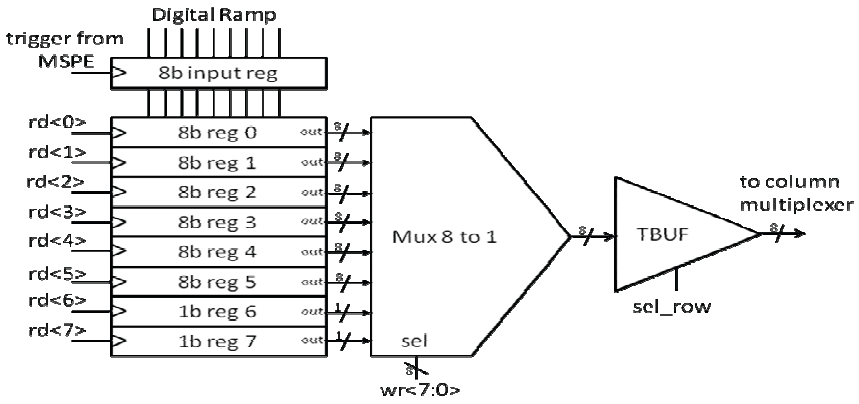

Figure 10. The proposed register bank and its content of the frame buffer for a single mixed-signal processor

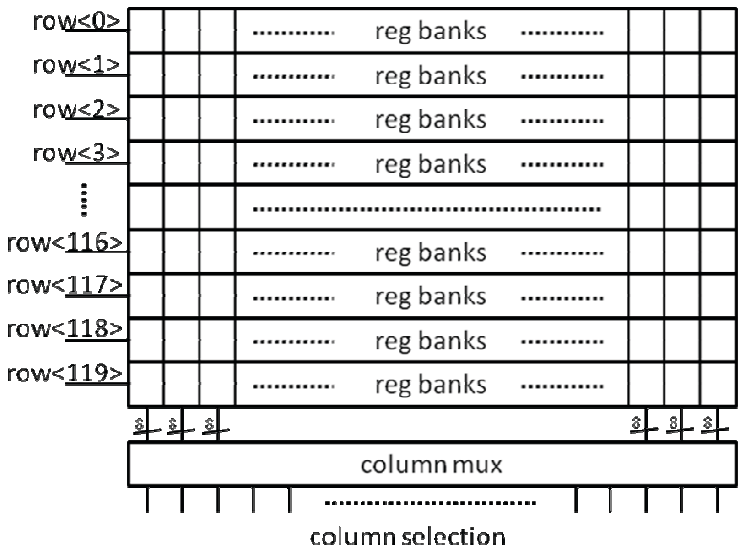

Figure 11. Figure shows register bank array and the column multiplexer.

The framebuffer array read-out is random access, the output data are packaged in 32-bits word. The data of whole row are sent to the output column multiplexer. The column multiplexer selects which columns are read. To read another row is necessary to download a complete row to the column multiplexer. The framebuffer can be programmed to read different image format.

7 DIGITAL PROCESSORS LAYER

The basic constructing element of our digital processor array architecture is the cell. The cells are locally interconnected. The processors in each cell can read the memory of their direct neighbors. There are boundary cells, which are relying data to handle different boundary conditions.

Each cell contains an arithmetic processor unit, a morphologic processor unit, data memory, internal and external communication unit. The arithmetic unit contains an 8 bit multiple-add core with a 24 bit accumulator, and 8 pieces of 8 bit registers. This makes possible to perform either 8,16, or 24 bit precision general calculations. The arithmetic unit can calculate multiplication, multiple-add, addition, subtraction, and comparison operation. Image processing primitives, like block matching, convolution, look-up table, diffusion, thresholding, rank order filters, contour detection, Sobel operator, median, etc. can be efficiently implemented by using the instruction set of the processor.

The morphology unit supports the processing of black-and-white images. It contains 8 pieces of single bit morphology processor, for parallel calculation of local or spatial logic operations, like erosion, dilation, opening, closing, hit and miss operations, etc. The detailed description of the processor array and the cells can be found in [10][10].

8 CONCLUSIONS

A new vision sensor-processor chip is introduced. The special feature of the new architecture is that it combines multiple cellular processor arrays operating in different signal domain, and processing the image in different scale and resolution. The new design is implemented by applying the latest 3D silicon integration technology.

9 ACKNOWLEDGEMENTS

This work has been partially funded by the ONR under Project VISCUBE and the Junta de Andalucía under Project VMOTE:2006-TIC-2352.

10 REFERENCES

[1] ENIAC working group, *Strategic Research Agenda* (2nd edition), European Technology Platform Initiative (2007).

[2] 2007 International Technology Roadmap for Semiconductors (ITRS) 2007 Edition Emerging Research Devices. http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[3] A. El Gamal and H. Eltoukhy, "CMOS Image Sensors," IEEE Circuits and Devices Magazine, 6-20 (2005).

[4] G. Devaraj et al., "Applying Algorithms," Vision System Design 13(11), 17-20 and 85-87(2008).

[5] A. Rodríguez-Vázquez et al., "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs," IEEE Transactions on Circuits and Systems-I, 51(5), 851-863 (2004).

[6] G. Liñán et al., "A 1000FPS@128x128 Vision Processor with 8-Bit Digitized I/O, " IEEE Journal of Solid-State Circuits, 39(7), 1044-1055 (2004).

[7] R. Carmona et al., "A Bio-Inspired 2-Layer Mixed-Signal Mixed-Signal Flexible Programmable Chip for Early Vision," IEEE Transactions on Neural Networks, 14(5), 1313-1336 (2003).

[8] A. Rodríguez-Vázquez et al., "A CMOS Vision System On-Chip with Multi-Core, Cellular Sensory-Processing Front-End," [*Cellular Nanoscale Sensory Wave Computers* - edited by C. Baatar, W. Porod and T. Roska]. Springer 2010.

[9] J.C. Russ, [The Image Processing Handbook]. CRC Press 1992.

[10] P. Földesy et al., "Configurable 3D Integrated Focal-Plane Sensor-Processor Array Architecture," Int. J. Circuit Theory and Applications, 573-588 (2008).