VISCUBE: a multi-layer vision chip

Ákos Zarándy*[#], Csaba Rekeczky[#], Péter Földesy*[#], Ricardo Carmona-Galán[§], Gustavo Liñán Cembrano[§], Gergely Sós*, Ángel Rodríguez-Vázquez[+], Tamás Roska*

[#]Eutecus Inc, Berkeley, CA, USA

*MTA-SZTAKI Budapest, Hungary

[+]AnaFocus, Seville, Spain

[§]Instituto de Microelectrónica de Sevilla, (IMSE-CNM), Sevilla, Spain

Abstract Vertically integrated focal-plane sensor-processor chip design, combining image sensor with mixed-signal and digital processor arrays on a 4 layer structure are introduced. The mixed-signal processor array is designed to perform early image processing, while the role of the digital processor array is to accomplish foveal processing. The architecture supports multi-scale, multi-fovea processing. The deign was made for MIT Lincoln Laboratory 3DM2 0.15micrometer technology.

1 Introduction

The main trade-off of the focal-plane sensor-processor design considerations is the silicon resource distribution between the sensors and the processors. The problem already starts at the technology selection. Conservative technologies with thicker depletion layer and fewer metal/dielectric layers have larger light response; hence they are better suited to be sensor materials. However, processor circuits can benefit from modern deep sub-micron technologies because both the large number of metals and the small minimal size transistors needed to build complex circuits, especially in the digital domain. After the technology is selected, the next hard trade-off is the relation of the sensor area and processor area. The minimal sensor area is determined by the sensitivity requirements, while the fill factor and the maximum chip area limit the processor size. If we cannot squeeze the processor circuits to the required area, we need to choose a finer silicon technology, and the iteration starts again.

Vertical integration technology [4] makes possible the breakout form this circle. It enables the integration of multiple silicon and/or other semiconductor layers above each other by interconnecting them with *thru silicon vias* (TSVs) [4]. In this way, the sensors may occupy the entire top layer, leading to 100% fill factor [5]. Moreover, besides the visual domain, IR image sensor can be implemented also by applying InGaAs [6] or other materials. Analog or mixed-signal layer for signal conditioning, early image processing and AD conversion can be implemented on a different silicon layer below, and digital post processor layer(s) can be built even below. In ideal case, we can even choose different silicon technology for the different layers, which are well suited to the requirements of the implementation of the optimal sensor, analog/mixed-signal cells, and digital processors.

This chapter introduces a focal-plane sensor-processor circuit design which uses vertical integration technology. The work was done in the framework of the VISCUBE project [1], which is led by Eutecus Inc. Berkeley, California, and financed by the Office of Naval Research (N00173-08C-4005). Two European design groups are involved in the VLSI design, one from Spain (AnaFocus LLC), and one from Hungary (MTA-SZTAKI). The goal of the project is to build a visual surveillance, reconnaissance, and navigation device, which can be carried by miniature *unmanned aerial vehicle* (UAV). Due to the limited payload and power supply on board, a single chip vision system, under 1W was the goal.

The algorithmic requirements are to perform moving platform video analytics, including feature point extraction, displacement calculation (optical flow), and feature extraction in multiple windows [3]. The image capturing and processing speed should have to reach 1000FPS. To be able to fulfill these goals, the design supports multi-scale multi-fovea processing.

The chapter is organized on the following way. After the introduction, the architecture and the technology are introduced. It is followed by the implementation details. Then, the simulated operational examples, and finally, the conclusions are shown.

2 Architecture

In general, the architecture is derived by the required functionalities and the available technologies. Here the required functionalities are

- Image sensing;
- Moving platform image analysis:
  - Feature point extraction;
  - Displacement calculation;
  - Feature extraction;
- Analog to digital conversion.

To fulfill these computational requirements in the given strict power budget, we have to apply advanced algorithmic approaches such as multi-scale and multi-fovea techniques. The common feature of these techniques is that they significantly reduce the amount of data to be processed. The multi-scale algorithms apply subsampling, hence lower resolution images are processed only.

Fovea processing techniques use detailed image analysis on small windows of the image only. When these two methods are combined, a low resolution version of the image is fully preprocessed, and as a result of this, the areas of interests are detected. These areas of interests (windows) require further detailed processing. The size and the resolution (scale) of these windows depend on the type of analysis applied.

When selecting image processor devices we need to study the efficiency of different architectures. For early image processing the mixed-signal locally interconnected processor arrays and the digital pass-through (pipe-line) pixel flow processor arrays can be efficiently applied [15]. However, above 1000 FPS under low latency requirements in near-sensor processing, the fine-grain mixed-signal processor array is the best choice. For fovea type post processing, coarse-grain digital processor arrays can provide efficient solution [15].

To connect and efficiently use the fine-grain mixed-signal processor array and the digital foveal processor array, we need (i) analog to digital converters; (ii) a digital frame buffer, which can store the entire frame; (iii) and a switch matrix, which makes possible glue-less scaling and windowing, to prepare data for the multi-scale foveal processor array.

In this design, we consider three scales (Fig. 1):

Scale 0:	Sensor resolution, 320x240 grayscale, 8-bit representation
Scale 1:	Mixed-signal processor resolution, 160x120 grayscale or binary
Scale 2:	Downscaled Scale 1 resolution, 80x60 grayscale, 8-bit

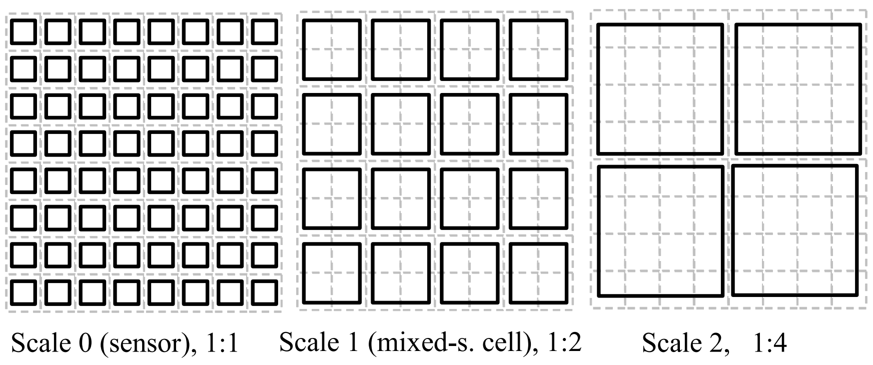

Fig. 1. The three different scales used in the VISCUBE design

2.1 Technology

The targeted technology contains 3 *silicon on insulator* (SOI) layers 0.15 micron with 3 metals (plus backmetal on tier 2 and tier 3) provided by MIT Lincoln Laboratory (Fig. 2) [8]. The layers are interconnected with 5 micron pitch Tungsten *thru silicon vias* (TSV). The back-illuminated sensor array is implemented on an additional bump bonded semiconductor layer on top.

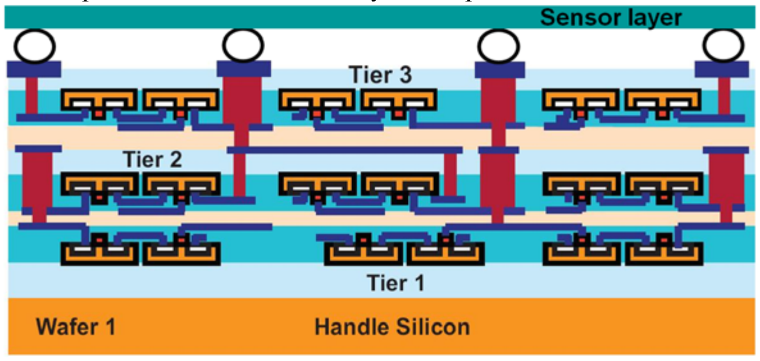

Fig. 2. Targeted vertical integration technology whit 3 SOI layers and a bump bonded sensor layer. (MITLL 3DM2 0.15) [8]

2.2 Architecture mapping

After defining the necessary architecture components, and having seen the 4 layers provided by the technology, the architecture mapping is straightforward (Fig. 3). The mixed-signal layer needs to be next to the sensors, because it processes the raw analog sensor data, and takes part in the AD conversion. Between the mixed-signal and the foveal processing layer, the frame buffer and the switch matrix takes place, to provide efficient data communication.

Sensor array
Mixed-signal processor array + AD converter
Frame buffer + switch matrix
Foveal processor array

Fig. 3. Architecture mapping

After the architecture mapping is described, we introduce the architecture of the individual functional blocks and layers. The implementation details are introduced in Section 3.

2.3 Sensor layer

As it was said already, the back-illuminated sensor array is built on an extra semiconductor layer, connected to the processor array using bump bonding interface. The resolution of the sensor array is 320x240. The choice of the sensor material defines the wavelength. This can range from visual range to NIR, or even SWIR. The sensor layer contains the photodiodes only. The integration type sensor interface, implemented on the mixed-signal layer, keeps a constant bias voltage on the diodes. This supports the usage of a wide range of sensor material, independently of their sensitivity characteristics.

2.4 Mixed-signal processor array

The fine-grain mixed-signal processor array occupies the top silicon layer. Its resolution is 160x120 (Scale 1). The *mixed-signal processing elements* (MSPE) serve the image acquisition; they can calculate diffusion, subtraction; and identify the local maxima. *Difference of Gaussians* (DOG) filter can be calculated by subtracting different linear diffusion results of the same image.

Each MSPE (Fig. 4) contains (i) electrical interface for 4 photodiodes (4 pixels), 4 *local analog memories* (LAMs) for storing an entire Scale 0 image; (iii) an analog diffusion unit, (iv) a comparator, which is used either as the local extremum detector, or as a component of a single-slop AD converter. Each MSPE is interconnected with its 8 neighboring MSPEs, allowing for programmable real-time spatial processing operations.

The mixed-signal processor array is designed to be able to handle both 160x120 and 320x240 sized images. It can process 160x120 sized images. The spatial operators are designed to handle full Scale 1 images. The 160x120 sized image is generated by subsampling or binning the 320x240 image. The AD converter can convert either Scale 0, or Scale 1, or Scale 2 images.

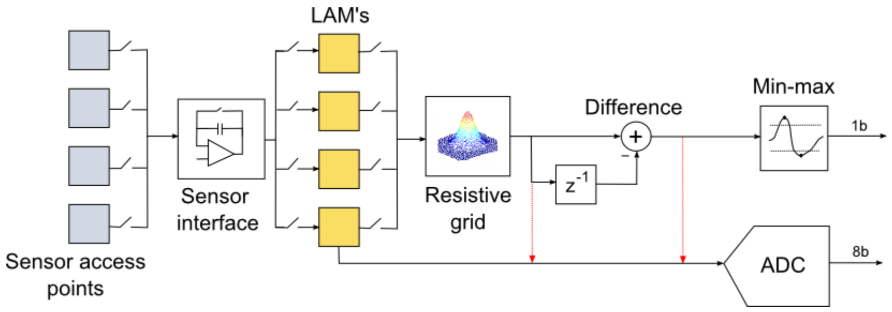

Fig. 4. MSPE architecture of the mixed-signal processor array layer

2.4.1 Image acquisition

Image acquisition is performed together by the sensor layer and the mixed-signal layer. The photodiodes are located on the sensor layer. Their photocurrent is integrated by the mixed-signal layer. The integration time is controllable in a wide range from sub-microsecond to hundred millisecond. The sensor interface applies a transconductance amplifier. The photodiodes are kept on a constant bias voltage during the integration. This bias voltage can be set externally to support various sensor materials.

2.4.2 Diffusion operator and DOG filter

The diffusion operator processes Scale 1 resolution (160x120) images. It is implemented with a discrete time 4 connected *switched capacitor* (SC) circuit which makes possible fine control on the diffusion scale. The diffusion process can be sampled and then continued, to be able to generate different images with different smoothing operators. By subtracting these images from each-other, we get difference of Gaussians filter.

2.4.3 Extremum location detection

The extremum location detection is also defined on Scale 1 grid. The output of the operator is a binary image, which is black in those pixels, which contains local maximum/minimum in a 3x3 neighborhood. In order to suppress irrelevant local extrema, a maximum/minimum pixel is accepted if its value exceeds a programmable global threshold limit.

2.4.4 AD conversion

Each of the MSPEs contains a single slope AD converter (Fig. 5). It contains four components (i) a digital to analog (DA) converter, (ii) comparator, a (iii) 8 bit counter, and (iv) a digital memory buffer. When an array of single slope AD converters are implemented, the DA converter and the counter can be shared, and only the comparator and the digital memory are needed to be built in each location. The comparator of the AD is located on Tier 3, in the mixed-signal layer, while the digital memory is located in the frame buffer layer on Tier 2.

This requires a pitch matched mixed-signal and frame buffer layer design. Each MSPE and its corresponding frame buffer cell, which contains the digital memory) is connected with one wire only (called trigger on Fig. 5).

Since there is one AD converter in each MSPE, the conversion of a Scale 0 image is done in 4 conversion cycles, while a Scale 1 image is converted in a single cycle. When a Scale 2 image is converted, every fourth AD converter is used.

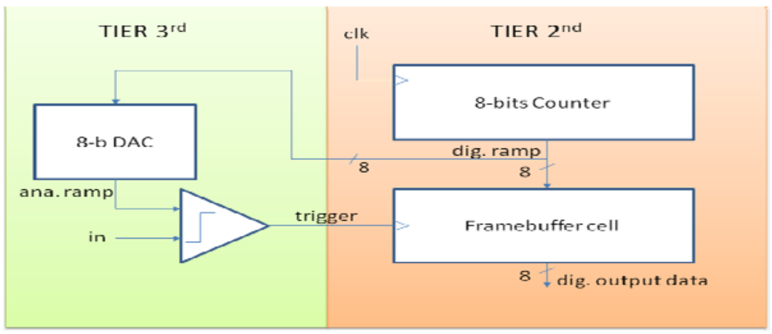

Fig. 5. Block diagram of the single slope AD. The comparator (left hand side) is implemented on the 3rd tier in every MSPE, while the digital register is located in Tier 2

2.4.5 Operation example of the mixed-signal layer

Fig. 6 illustrates the operation of mixed-signal layer. As it is shown it can capture a Scale 0 image, convert it to Scale 1 and Scale 2 images by applying either binning, subsampling or diffusion. It can also calculate difference of Gaussian, and can identify the local extreme points.

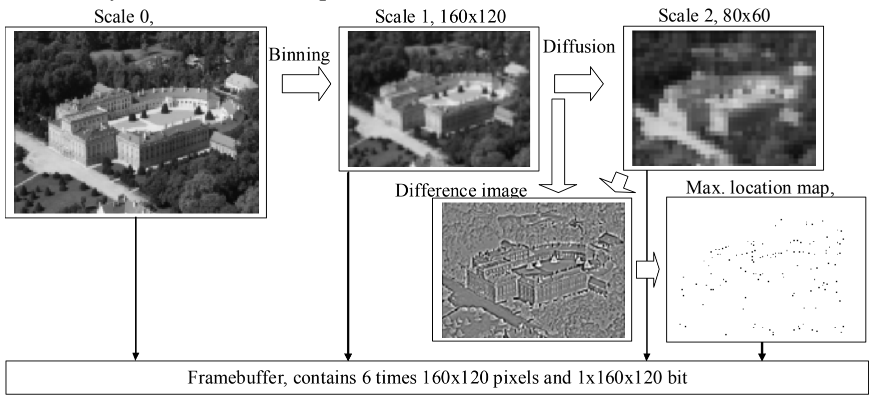

Fig. 6. Sample images of the algorithm executed on the mixed-signal processor layer (simulation)

2.5 Frame buffer layer

The frame buffer layer has three roles. It is used as the (i) digital registers of the single slope AD converters. It is also used as a (ii) storage and communication interface between the mixed-signal and the digital layer. Moreover, (iii) it supports random access to scaled or not scaled windows of the captured and preprocessed images, which is critical in the multi-scale and fovea processing approaches.

The frame buffer layer is constructed of an array of 160x120 memory units (Scale 1). Each memory unit is corresponding to the mixed-signal processor geometrically located exactly above it.

Each memory unit contains 6 bytes of memory and 2 bits (Fig. 7). 4 bytes are needed to store a Scale 0 image, because each unit/cell handles 4 pixels. The remaining 2 bytes can be used to store multiple downscaled images. The single bits can store the outputs of the extremum filters.

The registers of the frame buffer are constructed of twoport memories. They are written when an AD conversion or extremum operation is executed. They are read out by the digital processor array layer through a multiplexer. The multiplexer supports automatic windowing and downscaling functions to minimize IO time.

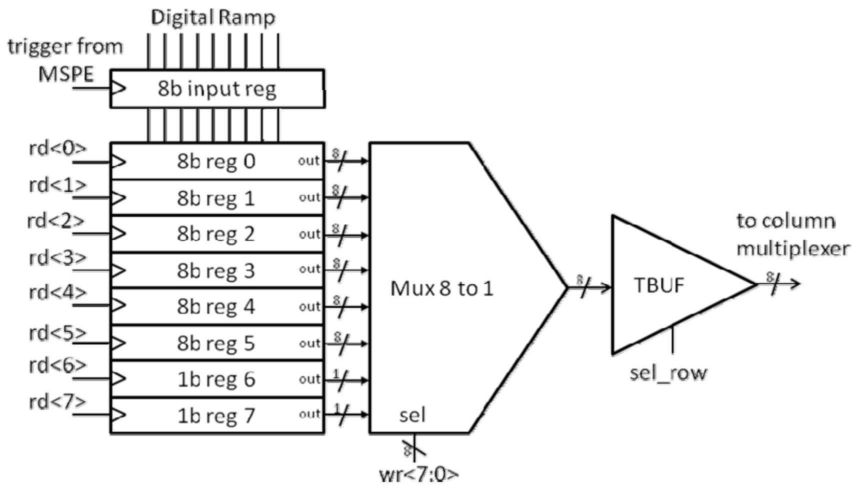

Fig. 7. Memory unit of the frame buffer layer

2.6 Foveal processor array

The foveal processor array is intended to be used for both area of interest (window) and full frame processing as well. This 8x8 digital processor array (Fig. 8) is an advanced version of our previous deign [16]. The distinguishing feature of this new implementation is the increased memory size and higher flexibility in the processed window size and distribution.

The processors can work in two different modes. In the first mode, the 64 processors are joining forces, and process a large or medium sized image (160x120, 80x60, or 64x64) by topographically distributing the image data among the processors. The second mode is used, when we have to execute the same operation (e.g. displacement calculation of a feature point) on a large number of smaller windows (24x24 or 16x16). In this case each processor processes one window individually. In the formal mode, the neighboring processors are exchanging data intensively, while in the latter mode, the processors are uncoupled. In both modes the processor array operates in *single instruction multiple data* (SIMD) mode.

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

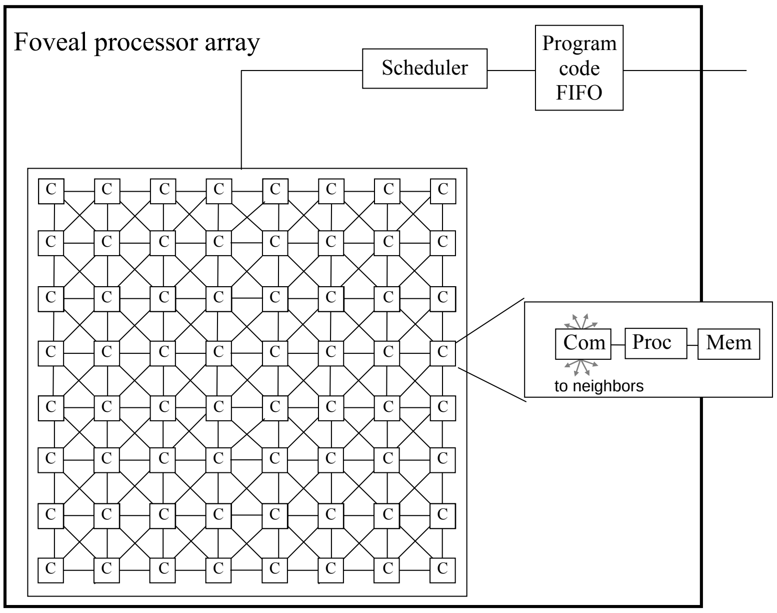

Fig. 8. The architecture of the foveal processor array

The block diagram of the processor kernel is shown in Fig. 9. 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 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 on the arithmetic processor.

The morphology unit supports the processing of black-and-white images. It contains 8 pieces of binary morphology processors, for parallel calculation of local or spatial logic operations, pixel-by-pixel binary logic, like erosion, dilation, opening, closing, hit and miss operations, etc.

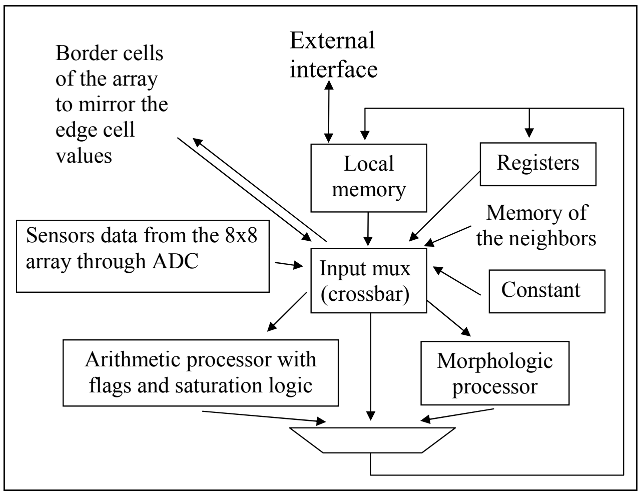
External interface
Border cells of the array to mirror the edge cell values
Local memory
Registers
Memory of the neighbors
Sensors data from the 8x8 array through ADC
Input mux (crossbar)
Constant
Arithmetic processor with flags and saturation logic
Morphologic processor

Fig. 9. The block diagram of the processor kernel

2.7 Control and communication

The VISCUBE design does not include internal control processor. This role can be played by an external control processor. The control processor can execute the main program of the system. It is responsible to initialize subroutines on the individual processor array layers, and image data communication among the three processing units. The control processor continuously evaluates the captured and preprocessed image flow arriving from the mixed-signal layer, and decides which parts (windows) of the input image requires more detailed analysis, and orders the digital processor array to accomplish the necessary routines. It is also responsible to switch between distributed or window processing modes, or modify parameterization of the routines according to the input image content.

The interconnection scheme of the main blocks is shown in Fig. 10. The sensor array, the mixed-signal processor array, and the memory units on the frame buffer layer are pitch-matched, hence direct parallel interconnections are used among them, utilizing the through silicon vias. In this way 4 sensor units are connected to one *mixed-signal processing element* (MSPE) cell. The interconnection between the MSPEs and the memory units are one-to-one. There are no parallel interconnection between the frame buffer and the foveal processor array. These layers are connected through a multiplexer and a fast digital bus only, because the windowing technique does not need full parallel interconnection topology.

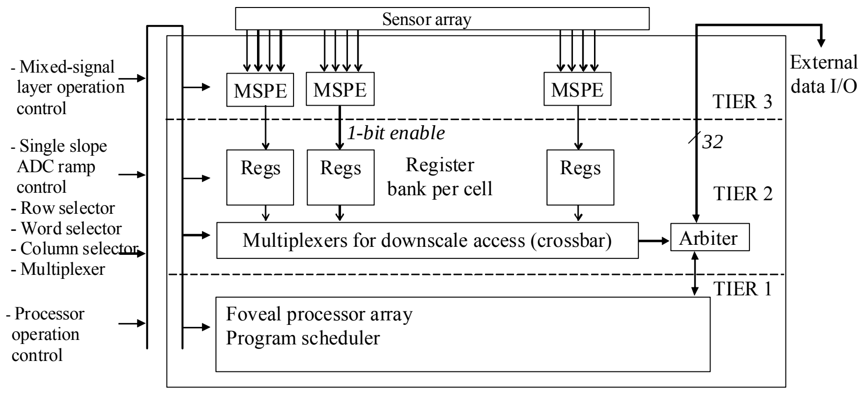

Fig. 10. The interconnection and the control scheme of the VISCUBE architecture.

3 Implementation

After the description of the architecture, the implementation is detailed in this section. In the following, the circuits on the three active tiers are described.

3.1 Tier-3: Mixed-signal processor layer

The entire layout of Tier-3 is shown in Fig. 11. As it is shown there, the mixed-signal processor layer has a topographic structure. Indeed, it is constructed of an array of 160x120 *mixed-signal processing elements* (MSPE) (Fig. 12). The size of an MSPE is 50x50 micron.

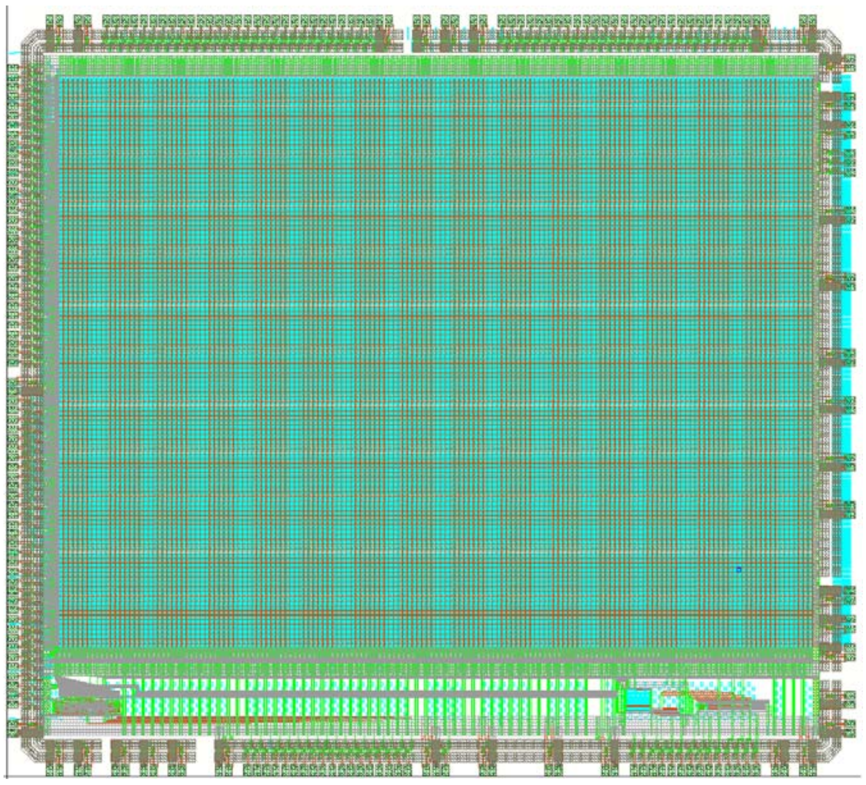

Fig. 11. Layout of Tier-3. It is constructed of a 120x160 sized, topographically arranged mixed-signal cell array (middle), and buffers, analog reference generators, a ramp generator, a programmable control unit, and IO circuits on the periphery.

Each MSPE (Fig. 4) includes four diodes/sensors interface circuits, 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 DOG filters, a local minimum and maximum locator and a single-ramp Analog Digital Converter.

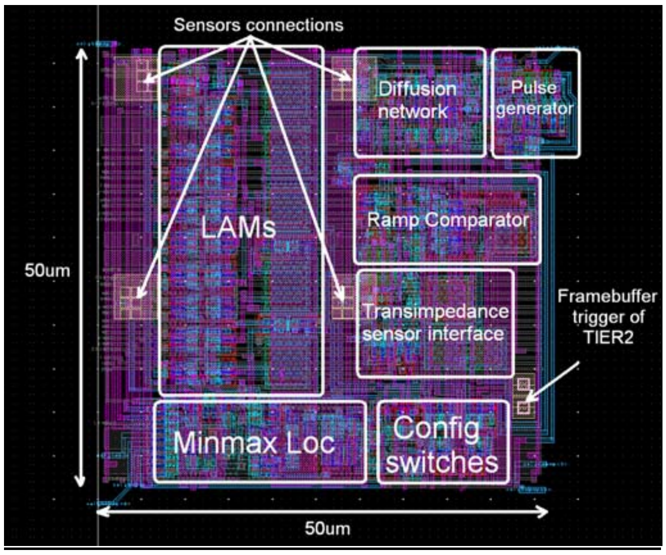

Fig. 12. Layout of the mixed-signal processor cell

3.1.1 Sensor Interface

Each of the processing elements of the mixed-signal array (160 x 120) provides access points for 4 diodes/sensors. The sensor interface has two stages, namely: a front-end composed of a *capacitive transimpedance amplifier* (CTIA) with offset sampling, and a *switch capacitance* (SC) circuit where the input data is stored in voltage form. These analog storage places (the LAMs) are also used to realize some arithmetic operations on the pixels data, such as averaging, diffusion, and subtraction.

The *capacitive transimpedance amplifier* (CTIA) architecture consists of a discrete time integrator (Fig. 13). The CTIA integrates the I_{ph} photocurrent during the integration period (Φ_{int} is on) and provides the following output:

$$V_o = V_{init} + \frac{I_{ph} * T_{\mathrm{int}}}{C_{\mathrm{int}}} \tag{1}$$

which can be tuned by changing the integration time.

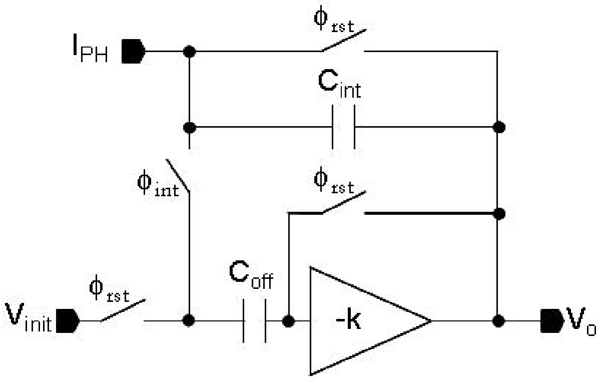

Fig. 13. Transimpedance sensor interface

LAMs are designed by using a sample and hold structure (Fig. 14), which contains some extra switches to perform arithmetic operations. Φ_{av01}, Φ_{av02} and Φ_{av23} switches are used to perform the averaging of the four memories containing the four sensor readings. Φ_{dif1} and Φ_{dif2} are used to connect independently the different capacitances with the diffusive network. Finally the switches Φ_{MT}, Φ_{MA0} Φ_{MA1}, Φ_{MA2}, Φ_{MB3} reconfigure the capacitors to calculate the subtraction of the stored data.

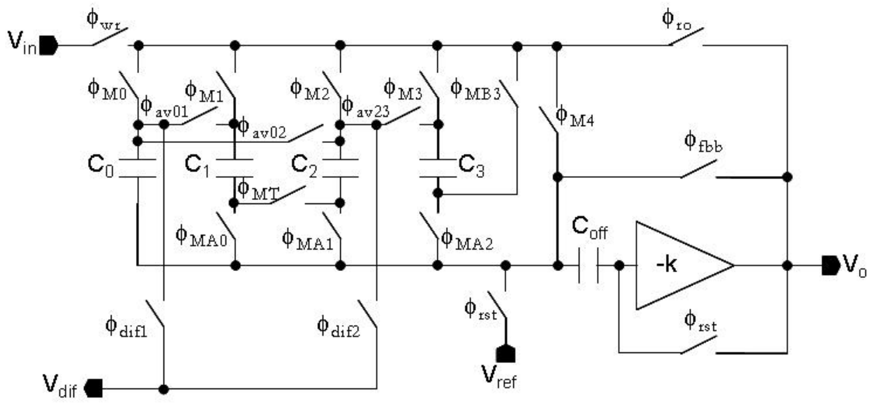

Fig. 14. Discrete-time sample & hold amplifier

3.1.2 Diffusion network for Gaussian pyramid generator

Gaussian filtering of the quarter image (160x120) is realized by a SC circuit. Fig. 15 shows the block diagram of the diffusion network unit. This circuit allows the transference of charge packages among nearest neigbours, thus emulating the behavior of a Gaussian filtering.

The image on the nodes is sampled at different point of the discrete time evolution. These sampling points correspond to particular scale representations. The accuracy of this correspondence is given by the performance of the SC emulator. The samples of the diffused image is either processed to calculate significant points on the Gaussian pyramid, or combined to generate the Laplacian pyramid, or converted to be stored in the frame buffer layer because of the needs of a particular algorithm.

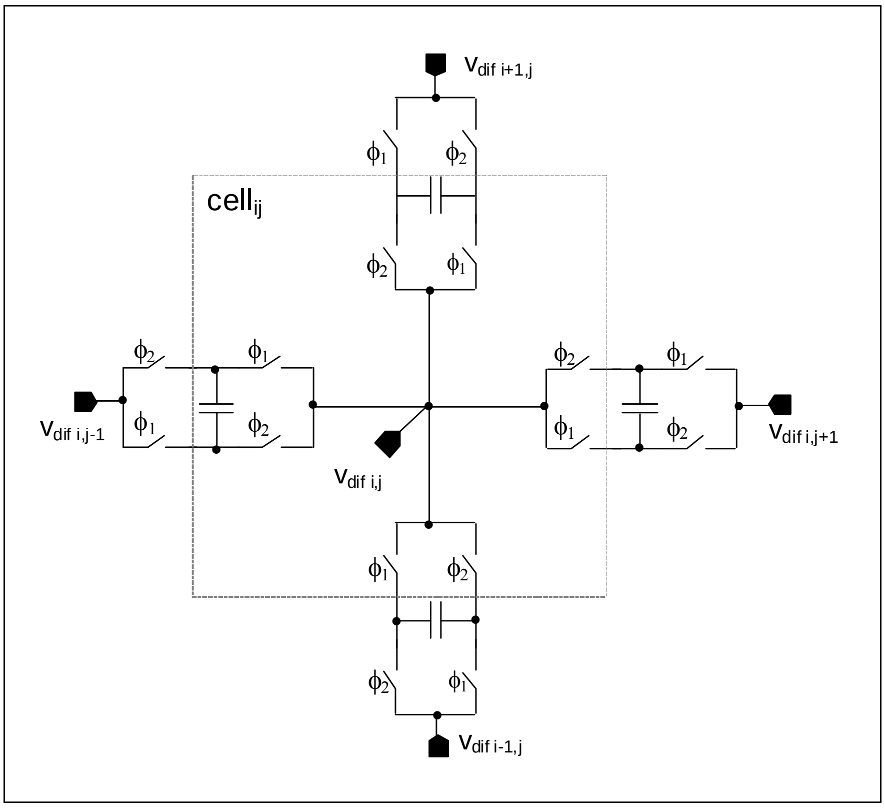

Fig. 15. SC Gaussian diffusion unit

3.1.3 Local extrema values detector

The local extrema value detector can identify the position of the local maxima or minima in 3x3 each neighborhood. In order to avoid the implementation of 8 comparators at each pixel the calculation is done in time, using the comparator and the ramp generator (DA converter) of the AD converter, a flip-flop and some logic circuit. The local minimum is calculated on a way that a rising ramp starts, and when it becomes larger than the value in a certain pixel position, it tries to set the flip-flop. However, the flip-flop can be set, if no other flip-flops in the 3x3 neighborhood is set already. In this way, maximum one flip-flop is set in each 3x3 neighborhood, and this flip-flop belongs to the smaller pixel value in that neighborhood. Fig. 16 shows the block diagram of the circuits that are used to calculate local maximum and minimum respectively.

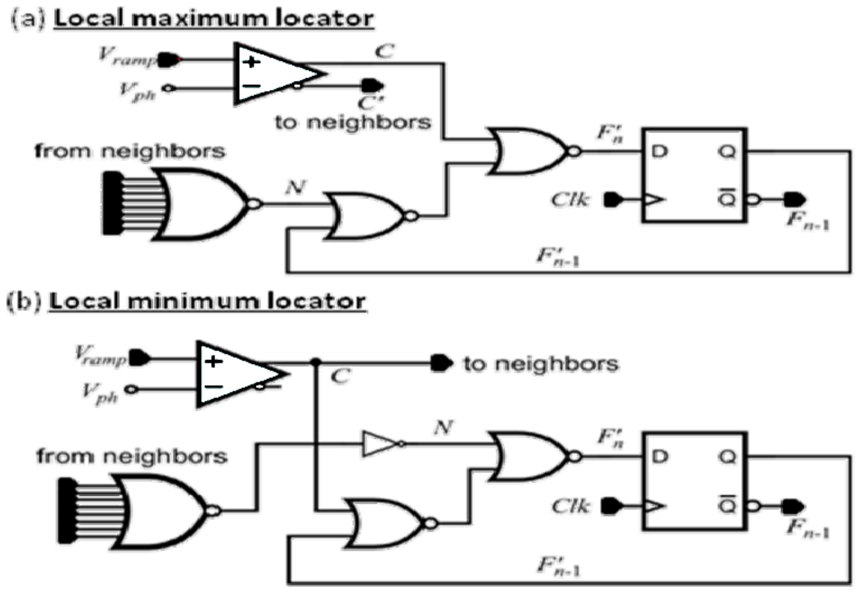

Fig. 16. - (a) local maximum locator (b) local minimum locator

3.1.4 AD converter and the comparator

The ADC is a per cell single slope converter. The comparator of the converter is in the 3rd tier, while the latch is implemented on the second tier (Fig. 5). The critical part of the design is the minimization of the offset variance of the large number of the comparators, because this directly leads to fix pattern noise. To achieve this, we used an offset compensation circuit in the comparator design (Fig. 17).

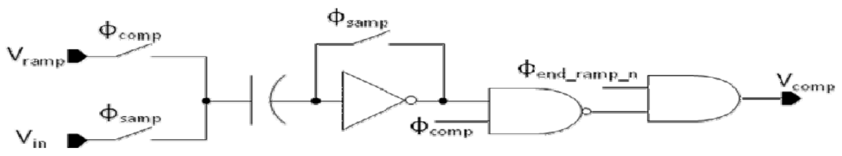

Fig. 17. Circuit diagram comparator

3.2 Tier-2: Frame buffer layer

The layout of the frame buffer layer is shown in Fig. 18. Its main function is to provide connection between the mixed-signal layer and the foveal processor layer. It is constructed of an array of 160x120 memory units. The array of the memory units are designed on a pitch matched way with the mixed-signal processor array, and the corresponding units are above each other, to be able to interconnect the two units with a TSV. This condition determines the size of the memory units to be 50x50 micron.

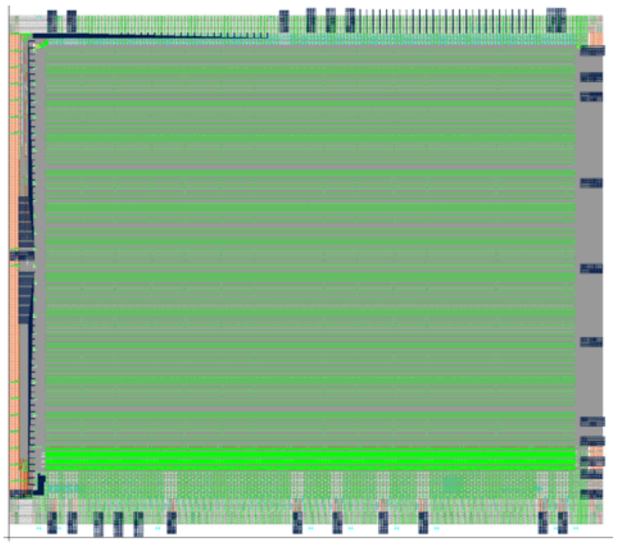

Fig. 18. Layout of Tier-2

The internal structure of the memory units in the frame buffer is derived from the storage and special accessing needs of the memory content. Each memory unit contains 6 bytes and 2 bits, plus the input register of the AD converter. This is 117kByte of memory, which can store 1 piece of Scale 0 image, 2 pieces of Scale 1 grayscale and 2 pieces of Scale 1 binary images. Since the images does not have a predefined position, the images can be interchanged. In this way, one Scale 0 image can be replaced with 4 pieces of Scale 1 image, and on the same way, one Scale 1 image can be replaced with 4 pieces of Scale 2 images. The layout of a memory unit is shown in Fig. 19.

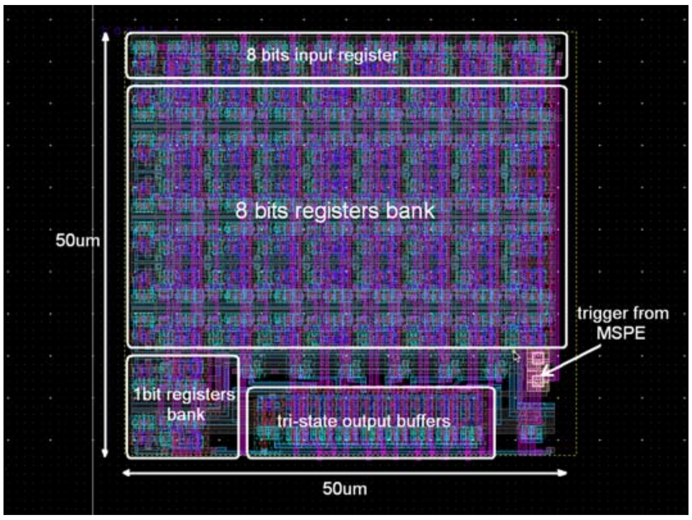

Fig. 19. The layout of a memory unit of the frame buffer

The output multiplexer of the memory units and the entire frame buffer is designed to efficiently serve the arbitrary window access demands of the foveal processor array. The frame buffer read-out is random access, the output data are packaged in 32-bits word. During the readout the data of a whole row are sent to the output column selector crossbar. The column selector crossbar connects the 4 selected columns, to the 32 bit output bus. This bus delivers data to the foveal processor array, and provides external access to the frame buffer (Fig. 10). The block diagram of a memory unit is shown in Fig. 7, while the structure of the frame buffer, including the column multiplexer is shown in Fig. 20.

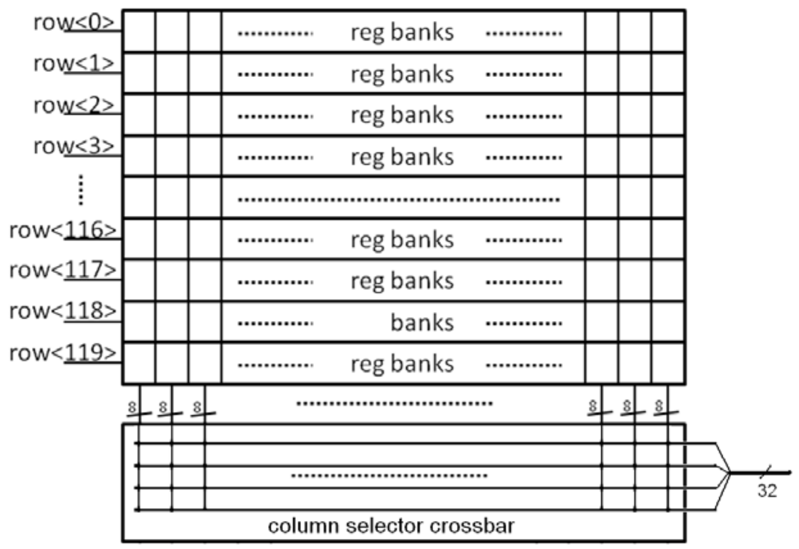

Fig. 20. Block diagram of the frame buffer and the columns selector crossbar

3.3 Tier-1: Digital processor layer

The primary role of the foveal processor array is to perform digital processing on either Scale 0, or Scale 1, or Scale 2 images or image parts. It is constructed of an 8x8 SIMD processor cell array (Fig. 8). The cells are locally interconnected.

Fig. 21 shows the layout the foveal processing array. The processor circuits are synthesized, while the memory blocks (the colored squares) are custom design. Neighboring memory blocks denoted with the same color belong to the same processor. The size of the memory blocks is 512 byte.

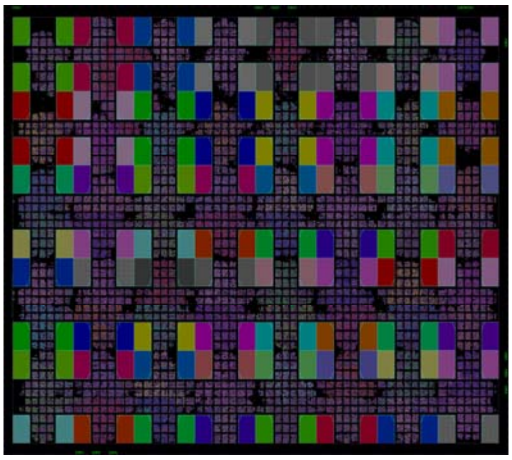

Fig. 21. Layout of Tier-1

3.3.1 Memory access

The communication between the processors is solved on a way that the processors can read the memory of their direct neighbors. This has been obtained by inserting an input multiplexer (crossbar) between the processor cores and their main memory (Fig. 9). The neighborhood memory access works concurrently all over the array, providing the avoidance of data congestion. The multiplexer is capable to access the neighboring memories and provide the required data. In this way, the processors need no distinguished instructions to operate on pixels that are actually stored in a neighbor processor. Furthermore, there are special boundary modules that are straightforward extensions of the arbiter mechanism. These modules provide the boundary condition of the array processor.

The upper two rows of processors (16 pieces) have 2 kbytes, while the lower 6 rows (48 pieces) have only 1 kbyte of memory. There is no pre-wired special purpose usage of the memory from hardware point of view, which means, that user may store any pixel of the images in any memory locations.

Two memory addressing types are supported. The first is the direct addressing mode, when all the processors access the same memory location in the same step. In this mode, the processors may access to their direct neighbors memory also. The second is the offset addressing method. In this mode, the content of one of the offset register is added to the global address coming from the scheduler. In offset addressing mode the memory of the neighboring processors cannot be accessed.

3.3.2 Image data distribution

There are to different operation modes of the processor array. The first is the topographic processing mode, when at least 32x32 sized images are cut to segments, and these segments are topographically mapped to the array. In this case, the neighboring processors read each-others memories to acquire pixel data for the neighborhood processing. The processor cell level memory requirements of the images are summarized in Table 1.

Image resolution	Size of a subimage handled by a cell	Memory requirements for gray-scale images bytes/cell
160x120 (Scale 1)	20x15	300
80x60 (Scale 2)	10x8	80
64x64	8x8	64
32x32	4x4	16

Table 1. Processor level memory requirements of an image in the digital processor array

The second operation mode is the non-topographic processing mode, when multiple (max 64) windows are cut out from a large image (typically Scale 0), and each of these windows are assigned to one processing cell. Since there are no independent program memories of the cells, the task to be completed should be the same, and cannot contain data dependent branches on the instruction level. However, thanks to the offset addressing mode, the algorithm may implement the same processing steps on different locations of the windows at the same time. This can be considered data dependent branch on the data level. This enables for example gradient based searches tasks, such as diamond search.

3.3.3 Arithmetic unit

The arithmetic unit contains an 8 bit multiply-add datapath logic with a 24 bit accumulator. The data path enables either 8 or 16 bit precision calculations. The arithmetic unit can calculate multiplication, multiple-add, addition, subtraction, and saturation operations (Fig. 22). Adopted from the common practice of handling both signed and unsigned data by the same unit, the hardware multiplier is of signed 9 by 9 bit precision, and the barrel shifter and the accumulator logic supports sign extension as well. The saturation mechanism also has a great importance in image processing, allowing the user to avoid the time consuming overflow and underflow management. Beside the arithmetical operations, this unit encompasses bit-field access as well.

The comparator unit is capable to evaluate the relation between signed or unsigned data of any modules. Depending on the outcome of the relation it sets several flags that are used in later operations.

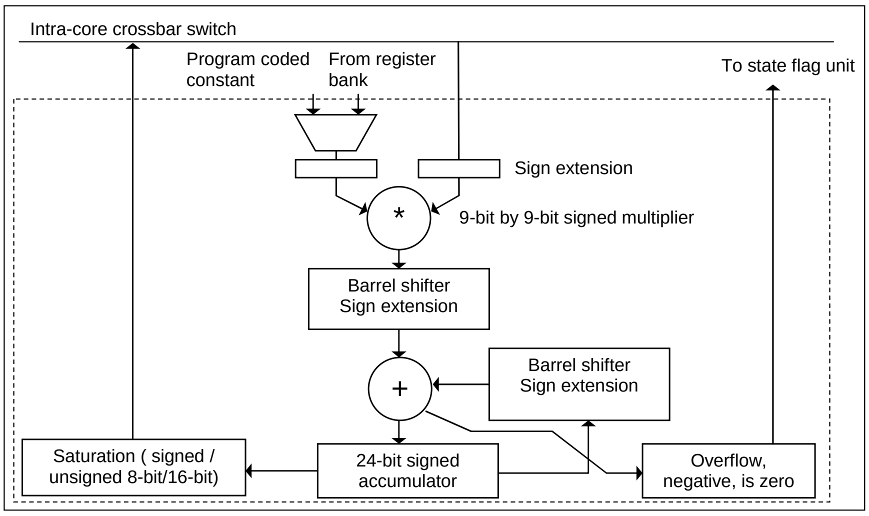
Intra-core crossbar switch
Program coded constant
From register bank
To state flag unit
Sign extension
*
9-bit by 9-bit signed multiplier
Barrel shifter Sign extension
+
Barrel shifter Sign extension
Saturation (signed / unsigned 8-bit/16-bit)
24-bit signed accumulator
Overflow, negative, is zero

Fig. 22. The architecture of the arithmetical units

3.3.4 Morphology unit

The morphology unit supports the processing of black-and-white images (i.e. the pixel representation is one bit per pixel). It contains eight pieces of identical single bit morphology processors (Fig. 23). Hence, it accelerates greatly the parallel calculation of local or spatial logic operations, like erosion, dilation, opening, closing, hit and miss operations. It can be efficiently used when each processor handles an 8 pixel wide image segment.

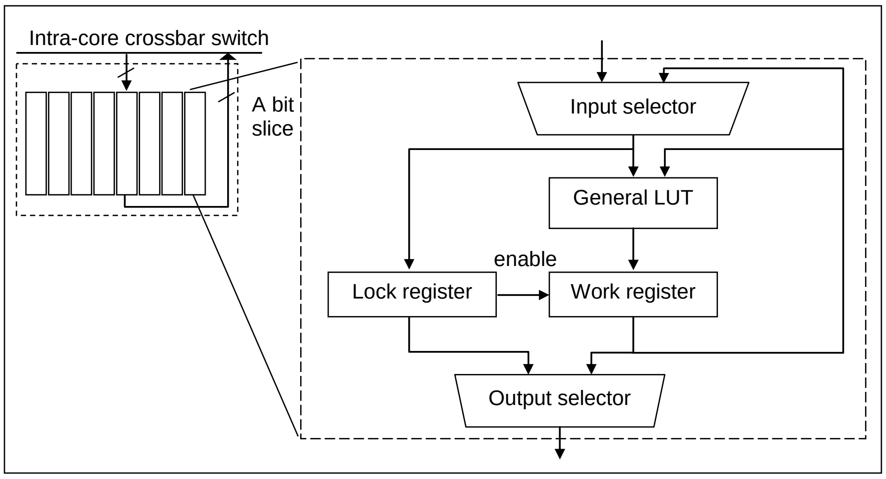

Fig. 23. The architecture of the binary morphology units

3.3.5 Scheduler

The cells do not have local program memory. The program is coming from a global off-array scheduler. Each processor receives identical command, parameters and attributes in each instruction cycle, which makes it a *single instruction, multiple data* (SIMD) processor array architecture. The individual processors are maskable. This means that content dependent masks may enable or disable the execution of a certain image processing operation in a few pixel locations. This masking can make the process locally adaptive.

The program scheduler receives the binary executable code from an on-chip FIFO. The depth of the FIFO is 16 words. When the program execution is enabled the scheduler takes an instruction out from the FIFO, decodes it, and sends it to the digital processor array. This process is going on as long as the execution is enabled and the FIFO contains code. This FIFO is fed from an external source. It uses the digital I/O bus of the system.

3.3.6 Instructions of the processor array

The instruction set of the digital processor array contains five groups:

- Initialization instructions
- Data transfer instructions
- Arithmetic instructions
- Logic instructions
- Comparison instructions

The Initialization instructions are needed to clear or set the accumulator, the boundary condition registers, the masks, and other registers of the cells.

The arithmetic instruction set contains addition, subtraction, multiplication, multiple-add operation, and shift. These operators certainly set the flags of the arithmetic units. These flags can be used in the next instruction as conditions.

The logic operations strongly support the execution of the binary mathematical morphology operations, like erosion and dilation. Since the internal operand width is 8 bit, each processor can handle 8 pixels in one clock cycle when executing logic operation.

The comparison instructions are introduced to calculate the relation between two scalars. These operators can be used for statistical filter implementations.

Using these instructions, we can efficiently implement the basic image processing functions (convolution, statistical filters, gradient, grayscale and binary mathematical morphology, etc) on the processor array.

4 Operational example: Displacement calculation

An operation example is shown in this section to demonstrate the operation of this Viscube architecture. In this example, we show how to calculate displacement for multiple points. The goal of the displacement calculation is to support image registration and optical-flow estimation also. In our UAV reconnaissance project, we assume high image sampling rate (500-1000 fps), hence the displacement and the rotation, introduced by the ego-motion of the on-board camera is supposed to be small.

Our strategy is to calculate the displacement of two consecutive images in 64 feature points. The displacements in the different positions may be different, due to (i) 3D structures in the ground, (ii) the rotation of the camera, and (iii) false measurement results. The displacement calculation starts with the identification of those points, where the displacement is calculated. These points are called feature points. Here we show two possibilities for the identification of the feature points, and the displacement calculation.

4.1 Identification of the feature points

We can define feature point as a point of low self-similarity. A point is considered to be feature point, if the contrast changes are high in different orientations, and its local interconnection topology is unique in small neighborhood. This uniqueness is important, because during the displacement calculation, we calculate matches around these feature points, and if there are points with similar topologies in the neighborhood, the displacement calculation will not be unambiguous. Typical situation is, when we are looking for matching on a straight line. In this situation, multiple matching positions can be found. This is called aperture problem (Fig. 24).

However, calculation of the uniqueness, which is practically an autocorrelation, is very expensive, hence it is not used. Rather than that, special locations (like local extrema or corners) are selected as feature points, with the assumption that the image is not periodical, therefore there are no similar locations in the neighborhood.

The mixed-signal layer of the VISCUBE chips is designed to identify the local extrema in different scales, which is considered to be the primary method of selecting feature points. However, we show a feature point selection method implemented on the digital processor array as an alternative option as well. In both cases, the feature point identification is done in 160x120 (Scale 1) resolution.

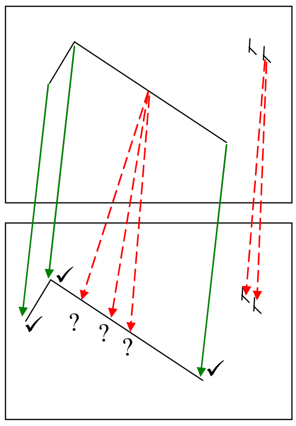

Fig. 24. Aperture problem in the displacement calculation. Two consecutive images with shifted pattern. The green solid arrow shows those locations, where the displacement can be calculated unambiguously, while the dotted red arrow shows a location, where the displacement is ambiguous

4.1.1 Local extrema identification using the mixed-signal processor layer

As we have seen, the mixed-signal layer is prepared to calculate *difference of Gaussian* (DOG) operators, and identify the local extrema positions on them. The variance of the Gaussians can be tuned, hence local extrema can be sought in different frequency components. To be able to reduce the number of the local extrema points, the minimal value of the maximum locations, and the maximal value of the minimum locations can be set. This avoids picking up the extrema coming from local noises in flat locations, which cannot be considered as feature points. The extrema calculation is done in 3x3 neighborhood. The DOG operation takes about 200 microseconds (sigma dependent), while the local minima and maxima calculation takes 25-25 microseconds.

4.1.2 Harris corner detection on the digital processor array

One of the most frequently used standard methods for identifying the feature points is the Harris corner detection [11]. Harris corner detector calculates the vertical and horizontal spatial derivatives of the image, puts them to the so called Harris matrix (2).

$$\mathbf{A} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{2}$$

where I_x and I_y are the spatial derivatives and the angle brackets denote averaging around the certain point (e.g. Gaussian averaging).

The special features of this matrix [11] are that

- if the eigenvalues are small, the area is flat;
- one large eigenvalue indicates an edge;
- two large eigenvalues indicate intersection of edges (corners, feature points).

However, the calculation of the eigenvalues is computationally expensive due to the square root. To be able to reduce the computational burden, it is enough to calculate (3) [12].

$$M_c = \det(\mathbf{A}) - \kappa\, trace^2(\mathbf{A}) \tag{3}$$

where κ is the sensitivity parameter. The value of κ has to be determined empirically, and in the literature values in the range 0.04 - 0.15 have been reported as feasible. The good feature points are indicated by the large values of M_c [12].

In the digital processor array, the 160x120 image is distributed among the processors, which means that each receives a 20x15 sized image. The image is smoothened first with an averaging filter and then, the spatial derivatives are calculated applying Sobel operators. After that, M_c is calculated, and in each 20x15 segment, the location of the largest M_c is selected to be the feature point. In this way the feature points are distributed over the image, and are not concentrated to the vertexes of a few high contrast objects.

The feature point search on an entire Scale 1 image takes 30,005 clock cycles, which is about 300 microseconds assuming 100MHz clock frequency.

4.2 Displacement calculation

The displacement calculation is done on a way that a certain location is cut out from the currently captured image around the feature point, and the best matching position is sought on the previous frame.

In our case an 11x11 window around the feature point is cut out from the original 320x240 image. In the first method, we assume that the displacement is not larger that 4 pixels, hence we are seeking the best matching position in a 9x9 window.

The matching is calculated by using L_2 *norm* (4)

$$L_2 norm = \sqrt{\sum_{x \in N} \left[I_n(x+h) - I_{n-1}(x) \right]^2} \tag{4}$$

where I_n are the intensity values in the current frame and I_{n-1} are the intensity values of the previous frame.

The best matching position is in that location, where the L_2 *norm* is minimal. (Certainly, the square root is not calculated.) To further reduce the computational needs, we do not calculate the L_2 *norm* in each location. There are several methods to reduce the number of the points, where the L_2 *norm* is actually calculated. These strategies assume that there is a monotonic decrease in the L_2 *norm* as we are proceeding towards the matching point.

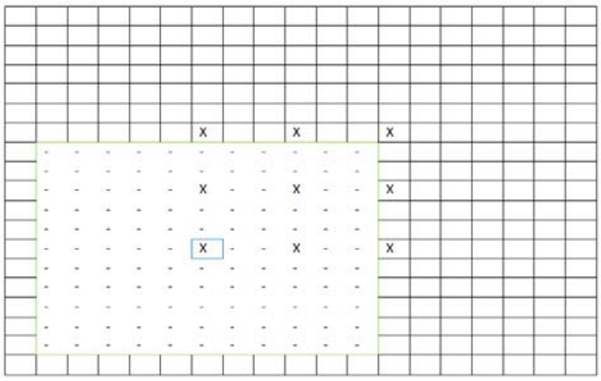

Fig. 25. Matching position searching strategy. The 11x11 sized window (indicated with dots) is cut out from the current image around the feature point. The large 19x19 window is cut out from the previous frame from the same location. The x shows those 9 positions, where L2 norm is calculated in the first step

In our case, first we calculate the L2 norm in 9 positions as it is shown in Fig. 25. Then, we perform 8 new calculations around the minimal value, and finally select the best one. This means that we perform 17 L_2 *norm* calculation altogether. This calculation takes 102,791 clock cycles, which is roughly 1ms.

In those cases, when the feature point was not strong (e.g. the entire 20x15 window was from the blue sky) we reject the displacement vector, because there is a high probability that is false. Fig. 26 shows an example of the identified displacement vectors.

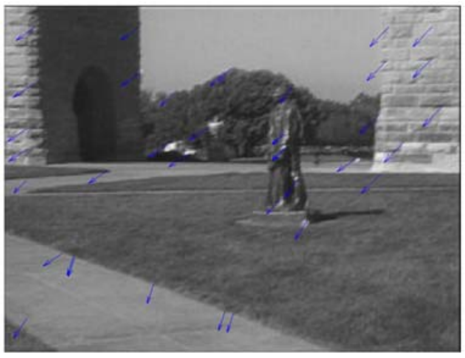

Fig. 26. Result of the displacement calculation

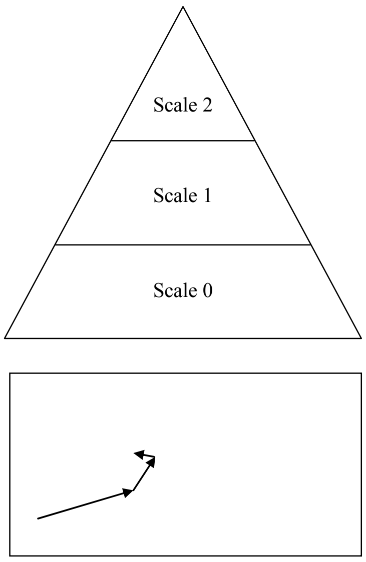

Fig. 27. Pyramid layers, and a possible sequential displacement calculation result.

If the displacement is larger than 4 pixels we have to use multi-scale (pyramid) method (Fig. 27). This means that we make the displacement calculation on different scales. It is calculated first on Scale 2. When the minimum of the L_2 *norm* is found, we start out the displacement calculation on Scale 1 from that point which was the calculated displacement (result) on Scale 2. After that, it is repeated for Scale 0 also on the same way. To calculate proper downscaling the diffusion operator of the mixed-signal layer is used. In this way the largest identifiable displacement goes up to 16 pixels. Fig. 28 shows a simulation example for the displacement calculation, where the pyramid method was used. As it can be seen both small and large displacements were identified.

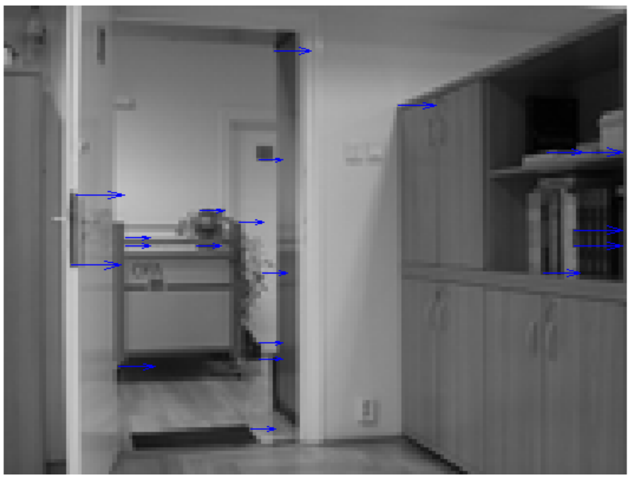

Fig. 28. Result of the displacement calculation on an image, where the displacements are larger than 4 pixels, hence pyramid methods applied

5 Conclusion

Advanced focal-plane sensor-processor design, based on vertical integration is introduced. The unique features of the design are that it incorporates heterogeneous processor arrays, one high resolution fine-grain processor array operating in the analog/mixed-signal domain, and one low resolution digital processor array. Algorithmically, the senor-processor architecture is designed to perform multi-scale, multi-fovea processing. The targeted application is UAV visual navigation and reconnaissance.

References

[1] Péter Földesy, Ricardo Carmona-Galan, Ákos Zarándy, Csaba Rekeczky, Angel Rodríguez-Vázquez, Tamás Roska: „3D multi-layer vision architecture for surveillance and reconnaissance applications", ECCTD-2009 Antalya, Turkey

[2] Peter Foldesy, Ricardo Carmona-Galan, Akos Zarandy and Csaba Rekeczky: „Digital processor array implementation aspects of a 3D multi-layer vision architecture", proceedings of the CNNA-2010, pp. 329-332 Berkeley, California, USA

[3] Akos Zarandy, David Fekete, Peter Foldesy, Gergely Soos and Csaba Rekeczky: „Displacement calculation algorithm on a heterogeneous multi-layer cellular sensor processor array", proceedings of the CNNA-2010, pp. 171-176 Berkeley, California, USA

[4] Philip Garrou, Christopher Bower, Peter Ramm, "Handbook of 3D Integration: Technology and Applications of 3D Integrated Circuits", Wiley-VCH, ISBN: 978-3-527-32034-9, 2008

[5] G. Deptuch, Vertical Integration of Integrated Circuits and Pixel Detectors, Vertex 2008Workshop, July 28-August 1, 2008, Uto island, Sweden.

[6] Marlon D. Enriquez, Michael A. Blessinger, Joseph V. Groppe, Thomas M. Sudol, Jesse Battaglia, Joseph Passe, Mark Stern, Bora M. Onat, "Performance of High Resolution Visible-InGaAs Imager for Day/Night Vision", Proc. of SPIE Vol. 6940, 69400O, (2008)

[7] S. Spiesshoefer, Z. Rahman, G. Vangara, S. Polamreddy, S. Burkett, and L. Schaper, "Process integration for through-silicon vias", J. Vacuum. Sci. Technol. A Vol. 23, Issue 4, pp. 824-829 (July 2005)

[8] MITLL Low-Power FDSOI CMOS Process Design Guide, 2006.

[9] A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, A. García, C. Utrera, M. Dolores Pardo, J. Listan, and R. Romay, "A CMOS Vision System On-Chip with Multi-Core, Cellular Sensory-Processing Front-End", in Cellular Nanoscale Sensory Wave Computing, edited by C. Baatar, W. Porod and T. Roska, ISBN: 978-1-4419-1010-3, 2009

[10] Takeo Kanade; Bruce D. Lucas, An Iterative Image Registration Technique with an Application to Stereo Vision; Computer Science Department Carnegie-Mellon University Pittsburgh, Pennsylvania 15213; From Proceedings of Imaging Understanding Workshop, pp. 121-130, 1981

[11] C. Harris and M. Stephens, "A combined corner and edge detector" (PDF). Proceedings of the 4th Alvey Vision Conference. pp. pp 147--151. 1988

[12] C. Harris, "Geometry from visual motion". in A. Blake and A. Yuille. Active Vision. MIT Press, Cambridge MA. 1992

[13] P. Foldesy, A. Zarandy, Cs. Rekeczky, and T. Roska, “Digital implementation of cellular sensor-computers”, Int. J. Circuit Theory and Applications (CTA), Volume 34 , Issue 4, Pages: 409 – 428, July 2006

[14] P. Foldesy, Á. Zarándy, Cs. Rekeczky, and T. Roska „Configurable 3D integrated focal-plane sensor-processor array architecture”, Int. J. Circuit Theory and Applications (CTA), pp: 573-588, 2008

[15] Akos Zarandy, and Csaba Rekeczky, “2D operators on topographic and non-topographic architectures -implementation, efficiency analysis, and architecture selection methodology”, *Int. J. Circ. Theor. Appl.,* 2010

[16] P. Földesy, Á. Zarándy, Cs. Rekeczky, and T. Roska „Configurable 3D integrated focal-plane sensor-processor array architecture”, Int. J. Circuit Theory and Applications (CTA), pp: 573-588, 2008