# Anatomy of the Focal-Plane Sensor-Processor Arrays

**Ákos Zarándy**

**Abstract** This introductory chapter describes the zoo of the basic focal-plane sensor-processor array architectures. The typical sensor-processor arrangements are shown, the typical operators are listed in separate groups, and the processor structures are analyzed. The chapter gives a compass to the reader to navigate among the different chip implementations, designs, and applications when reading the book.

## 1 Introduction

The spectrum of the focal-plane sensor-processor (FPSP) circuits is very wide. Some of them are special purpose devices, which are designed to optimally fulfill one particular task. A good industrial example for special purpose FPSP circuit is Canesta's depth sensor [1], which measures the depth information in every pixel based on the phase shift of a periodic illumination caused by the time-of-flight of the light. In our book, chapters by Fernández-Berni, Carmona-Galán, Posch, and Liñán-Cembrano et al. describe special purpose designs. The special purpose devices cannot be programmed, only their main parameters can be modified.

There are naturally general purpose FPSP devices also, which can be used in many different applications. A recently completed industrial general purpose FPSP chip is the Q-Eye, powering AnaFocus' Eye-RIS system [2]. These devices are fully programmable. This book introduces general purpose devices in chapters by Dudek, Laiho et al., Lopich, and Zarándy et al. (SCAMP-3, MIPA4k, ASPA, VISCUBE chips).

Other distinguishing feature is the domain of the processors. Some of the devices apply mixed-signal (partially analog) processors, while others use digital ones. The mixed-signal processors are smaller, consume less power, and do not require on-chip analog to digital converters. As a contrast, the digital processors are typically larger

Á. Zarándy (✉)
Computer and Automation Research Institute of the Hungarian Academy of Sciences, (MTA-SZTAKI), Budapest, Hungary
e-mail: zarandy@sztaki.hu

Á. Zarándy (ed.), *Focal-Plane Sensor-Processor Chips*,
DOI 10.1007/978-1-4419-6475-5_1, 

and more powerful, more accurate, and more versatile. In our book, mixed-signal arrays are shown in chapters by Dudek, Fernández-Berni, Carmona-Galán, and Posch (SCAMP-3, ATIS chips), while combined mixed-signal and digital processor arrays are shown in chapters by Laiho, Lopich, Dudek, Liñán-Cembrano et al., and Zarándy et al. (MIPA4k, ASPA, VISCUBE chips).

The type of the processed image is also an important issue. Some of the FPSP circuits are optimized for handling binary images only [3, 4]. Other devices can handle both grayscale and binary images. In our book, the general purpose SCAMP-3, MIPA4k, ASPA, and VISCUBE designs (chapters by Dudek, Laiho et al., Lopich, Dudek, and Zarándy et al.) represent this approach.

We have to distinguish the processing type also according to the neighborhood involved. The simplest operation is the pixel-wise processing, while the most complex is the global processing, where all the pixels in the frame are needed for the calculation as inputs.

This chapter is devoted to discuss the different architectural variances of the FPSP circuits. In the next session, various sensor processor arrangements are listed. This is followed by the description of the typical image processor operator types. Then, the processor architectures are shown. A more specific analysis of the operators and their implementations on these architectures can be found in [5].

## 2 Sensor-Processor Arrangements

There are two major components in all FPSP devices: the photo-sensor array and the processor(s). The number, the arrangement, the density, and the interconnection of these two components define the structure of the circuits. The aggregated computational capability of the processors and the processing needs on the data flow coming from the sensors are always balanced. In some cases, the number of the processors and the sensors are the same [2–4, 6, 7], in other cases; the number of the sensors is higher than the processors [8]. The sensors are typically arranged in 1D or 2D grids. These cases are discussed in the following subsections.

### *2.1 One-Dimensional Sensor Arrangement*

One-dimensional sensor (line sensor) is used when the objects or material to be captured are moving with a constant linear speed below the camera. Typical situations are the conveyor belt, or a scanning machine. The 1D arrangement is cheaper, because it uses smaller silicon surface. Moreover, higher spatial resolution can be reached (few thousand pixel wide image), and there is no boundary problem, which would come from merging individual snapshots. The chapter (***A Focal Plane Processor for Continuous-Time 1-D Optical Correlation Applications***) introduces a linear FPSP chip in this book.

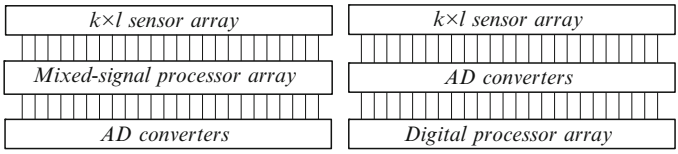


**Fig. 1** Typical 1D sensor processor arrangements with mixed-signal (*left*) or digital processors (*right*)

The one-dimensional sensor-processor arrays contain one or a few rows of sensors. In case of mixed-signal processors, the analog outputs of the sensors are directly processed. If digital processors are applied, analog-to-digital (AD) converters are needed between the sensors and the processor.

Figure 1 shows the typical 1D sensor processor arrangements. The sensor array contains one or a few long lines. The length of the lines can be from a few hundred to a few thousand pixels. Multiple lines are applied, if redundant or multi-spectral (e.g., color) information is needed.

In case of mixed-signal processor array, the number of the processors is typically the same as the number of the pixels in the sensor line(s), because the computational power and the versatility of these processors are limited.

In the digital version, the processors are more powerful and versatile. In this case, one or a few processor can serve the entire row. The number of AD converters typically matches with the number of digital processors.

## *2.2 Two-Dimensional Sensor Arrangement*

The versatility of the 2D arrays is larger than the 1D ones. It is worth to distinguish two basic types of arrangement from this family:

1. The sensor array and the processor array are separated. In this case, typically the size or the dimension of the sensor and the processor arrays are different.
2. The sensor array and the processor array are embedded into each other. This enables close sensor-processor cooperation, since the sensors and the processors are physically very close or directly next to each other.

These cases are detailed in the next two subsections.

### 2.2.1 Separated Sensor and Processor Arrays

One of the most critical parameters of the imagers is the spatial resolution. To be able to rich high spatial resolution, one needs to use small pixel pitch. The pitch of a sensor can be as small as a few microns, while a combined sensor-processor cell starts from 25 µm pitch on inexpensive planar technologies. Therefore, to be able

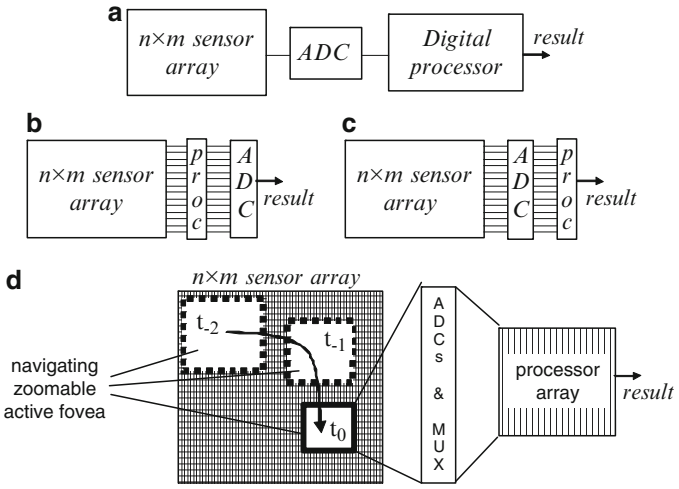


**Fig. 2** 2D sensor processor arrangements with separated sensor processor units. (**a**): one digital processor handles the entire image. Sensor arrays with mixed-signal (**b**) or digital (**c**) linear processor arrays. (**d**): foveal arrangement

apply high resolution (e.g., megapixel), one needs to separate the sensor array from the processors. The price, which is paid, is typically lower performance processing (speed or complexity), and/or reduced versatility.

We can see three different arrangements of the separated sensor-processor circuits in Fig. 2. In the first one, one digital processor serves the entire sensor array. The second variance applies linear column-wise processor arrays with mixed-signal or analog processor. The third one is the foveal arrangement. While in the previous two cases, the entire image is processed, in the foveal approach one or a few selected areas are involved into the calculations. This is an efficient approach if only some parts of the image carry relevant information to process. The description of the processor architectures and implementable operators are discussed in the next sections.

### 2.2.2 Embedded Sensor and Processor Arrays

Embedded sensor processor arrays are used when high speed is the critical parameter and not the high resolution. In this case above 10,000 visual decisions can be reached [9] in a second real-time even in complex situations on small or medium-sized images (<QVGA). We can distinguish two basic embedded processor types according to the processor density.

In the first case, there is a one-to-one correspondence between the sensors and the processors (Fig. 3). In this situation, a fine-grain processor array is used. These circuits typically apply either mixed-mode (SCAMP-3, chapter *SCAMP-3: A Vision Chip with SIMD Current-Mode Analogue Processor Array*), ACE16k [7], Q-Eye

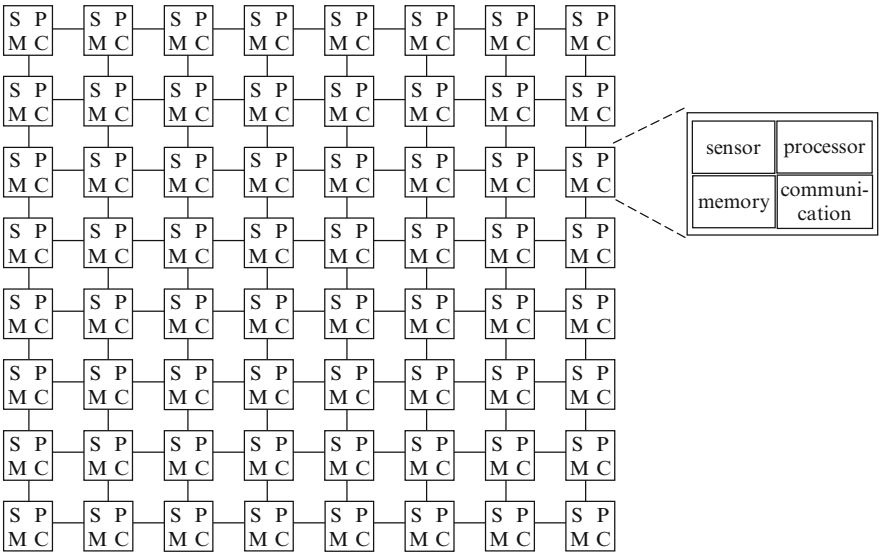


**Fig. 3** Embedded sensor-processor with fine-grain processor architecture

[2]) and/or 1 bit digital processors operating in bit-sliced mode (ASPA, chapter *ASPA: Asynchronous–Synchronous Focal-Plane Sensor-Processor Chip*; MIPA4k: chapter *MIPA4K: Mixed-Mode Cellular Processor Array*). The advantage of this structure is that it supports various efficient spatial–temporal operations such as diffusion, global OR, mean, address-event readout, etc. The description of these operations will be given in Sect. 4.3.2. Other advantage is that locally adaptive sensor control can be easily implemented such as locally changing exposure time according to illumination level or motion speeds [10, 11].

In the second case, a coarse-grain processor array is embedded to the sensor array [8]. This means that a $k \times k$ sub-array of pixels is assigned to one processor (Fig. 4). Naturally, more powerful, 8- or 16-bit digital processors are needed to process all the pixels. These processors can be more versatile than the mixed-signal or the bit sliced ones, and their communication radius is much larger, because they can reach the $k$th pixel in a single step. Besides the $k \times k$ array of pixels and the processor, each cell includes a memory and an AD converter. The size of the memory is typically enough for storing 6 or 8 pieces of the $k \times k$ image part. To squeeze the ADC to the limited area, one may use successive approximation type ADC or pixel-wise single slope one.

In both cases, the locality (mainly local data communication) plays important role. Thanks to this, these architectures are scalable, consume very low power, and suitable for further implementations with nanotechnology also, where the long communication lines are the main barriers.

Important issue of this technology is the sensor-processor tradeoff. On the one hand, both the sensor and the processors need relatively large area to be sensitive and to provide satisfactory computational power. On the other hand, the overall

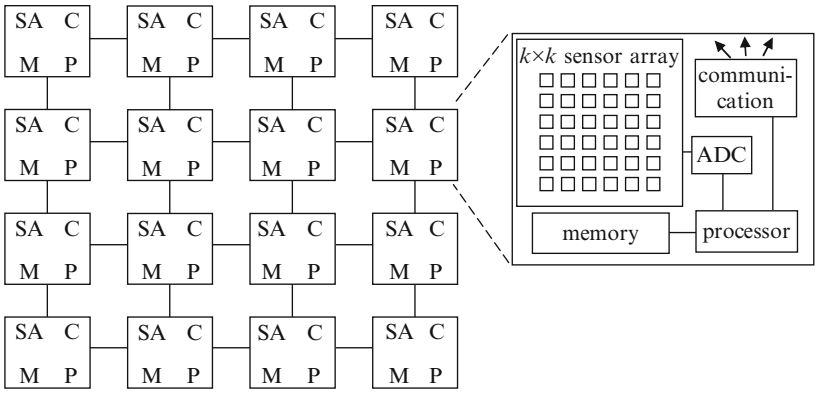


**Fig. 4** Embedded sensor-processor with coarse-grain processor architecture

area should be small; otherwise, the resolution will be too low. Moreover, the same silicon technology is not necessarily optimal for both the sensor and the processor circuits. The 3D integration can break through this bottleneck of the planar silicon technology. It introduces almost 100% fill factor, different sensor and processor materials, and extends the sensitivity gamut of the sensor. The chapter (*VISCUBE: A Multi-Layer Vision Chip*) of this book introduces a 3D sensor-processor design. Unfortunately, the 3D technology is not yet an established technology, hence it is still unreliable and expensive (year 2010). However it will change in a few years, and the 3D approach is expected to dominate the FPSP technology. A cheaper solution for increasing the fill factor nowadays without increasing the sensor area is the microlens technology [12].

## 3 Operator Types

A wide range of different operator primitives are applied in image processing and one can find many ways to classify them. Here, we apply two classification criteria (Fig. 5). The primary classification is done according to the output dimension (0D: scalar(s), 1D: row/column, 2D: or image). The third category (image → image) is further divided into three subcategories according to their relative input location and size, because for the locally interconnected processor arrays, the communication topology is one of the key properties. This section discusses these categories and lists the typical operators in the classes. The efficient implementation methods on different processor architectures will be described in the next section.

### *3.1 Image → Scalar(s) Operators*

The *image → scalar(s)* operators are typically used for feature extraction or localization. These operators can be implemented in a way that a processor scans the entire image, reading each of the pixel ones [5]. During this scan, some statistics

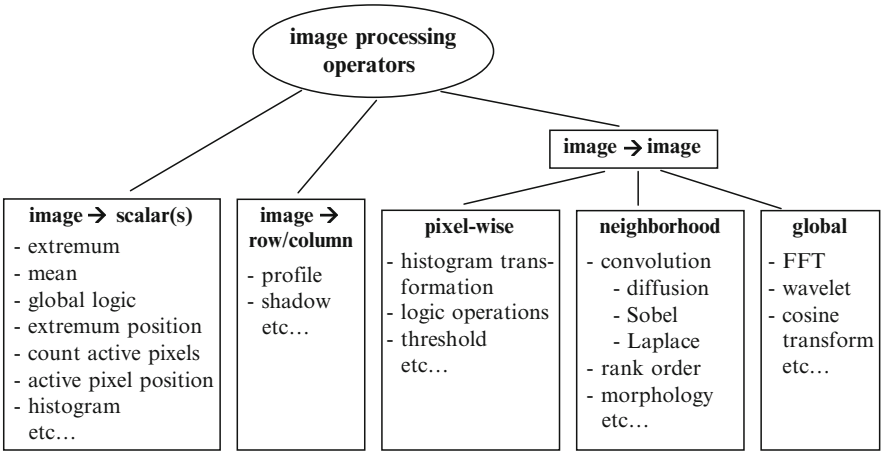


**Fig. 5** Operator classification

(min, max, mean, global OR, number of black pixels on a binary image, histogram, etc.) can be calculated. Similarly, the coordinates of extremum points or the white pixels on a binary image can be also calculated.

## *3.2 Image → Row/Column Operators*

The *image → row/column* operators apply 1D single scans row-wise or column-wise processing. In this case, the image lines or the columns are decoupled from each other, which means that the input domain of the operator is one line or one column. Typical examples here are the profile and the shadow operators.

## *3.3 Image → Image Operators*

The *image → image* operators are further divided into three subcategories. Some of these operators are defined for one input image (e.g., Sobel operation), others apply multiple input images (e.g., pixel-wise logic AND). In both cases, we examine here the input locality independently from the number of the input images.

### 3.3.1 Pixel-Wise

The *pixel-wise* operators include those operators, which require only the pixel itself as an input, and no information from the neighborhood is needed. These operators can be described in the following form:

$$y_{ij} = f(u_{ij}), \tag{1}$$

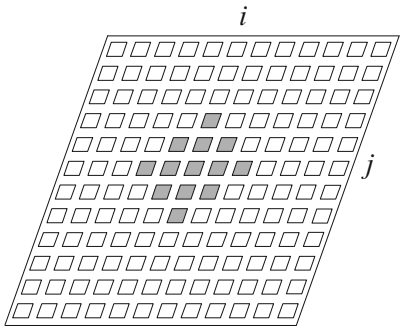


**Fig. 6** Input domain of a neighborhood operator

where

$u_{ij}$ is the input pixel;
$y_{ij}$ is the result pixel;
$f()$ is a one input scalar output function (assuming one input image).

Typical operators here are the gain, offset, and contrast manipulation (histogram transformation), and the thresholding.

### 3.3.2 Neighborhood Processing

The special feature of this class of operators is that the input is coming from a relatively small neighborhood of the pixel. Figure 6 shows an example, where 13 input parameters are used to calculate the operator in the $ij$ position. The neighborhood radius indicates the distance between the central pixel position and the farthest pixel in the input domain.

$$y_{ij} = g(U_{ij}), \tag{2}$$

where

$U_{ij}$ is the input domain;
$y_{ij}$ is the result pixel;
$g()$ is a multiple input scalar output function (assuming one input image).

In many cases, these operators are applied multiple times. These are called iterative calculations.

These operators are also called topographic operators, because they apply local operations on topographically mapped data sets.

### 3.3.3 Global Processing

The input domain of the global *image* $\rightarrow$ *image* operators are the entire image. Typical operators here are the Fast Fourier transformation (FFT), the wavelet transformation, or a Hough transform.

# 4 Processor Arrangements

Here, we list the processor structures used on FPSP chips and examine their operator execution capabilities with different memory sizes. Here, we consider the execution of one single operator primitive only. The efficiency figures of these architectures are calculated in [5].

## *4.1 Single Processor Architectures*

This processor arrangement is constructed of a processor and some memory (Fig. 7). It can be applied next to either a 1D or a 2D sensor array. The data-stream is coming from the sensor array sequentially. This means that the pixels are coming from left to right in a row, and the rows from the top to the bottom. The capability of the processor is defined by its memory size and its internal architecture. Here, we distinguish three different memory sizes, which are enough to store:

1. A few pixels.
2. A few lines.
3. A few frames.

These are discussed in the next subsections.

### 4.1.1 Single Processor with Small Memory

The simplest possible processor arrangement of an FPSP is the single processor unit with a small memory, which is enough to store a few pixels and some other data. They can execute *image* → *scalar(s)* operators and *pixel-wise* operators. Moreover, they can execute those *image* → *row* operators, which are row-wise and have left to right propagation direction, same as the pixel flow. For example, a vertical profile or a left to right shadow can be implemented, while neither vertical, nor right to left shadows can be.

In simpler case (e.g., gain or contrast modification, extremum finding), both mixed-signal or digital units can be used. More complex operators (e.g., histogram)

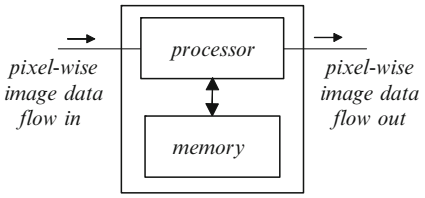


**Fig. 7** Single processor arrangement

require digital processor units. These processors are typically special purpose ones, where only the parameters and/or some arguments of the processing can be set.

### 4.1.2 Single Processor with Medium-Sized Memory

The memory size here is large enough to store a few lines from the image. This efficiently supports the execution of *image → scalar*, *pixel-wise*, and the *neighborhood* operators. Moreover, the *image → row/column* operators can be executed, where the propagation type is left to right or top to bottom. (The column wise operators can be executed also because entire row fits to the memory.) These operators are the vertical or horizontal profile and the left to right or top to bottom shadows.

Here, typical digital processor architecture is applied, because it requires complex memory management. The processor is still special purpose, with settable parameters/attributes. Since the processor receives sequential pixel stream, it cannot deal more with a pixel than the pixel clock period. This processor architecture can be efficiently used in video processing [13, 14], because most of the important operators can be implemented on them, but their memory is still small.

### 4.1.3 Single Processor with Large Memory

The simplest general purpose vision chip concept is to integrate a processor with large enough memory to store a few frames next to a sensor array. This type of processor can implement all kinds of operators, since it can access the entire image. In this case, typically fully programmable processors are applied, hence the vision chip.

The drawback of this kind of architecture is that a single processor can provide relatively small processing power, hence only simple or low frame rate applications are possible. Moreover, the sensor cannot be high (VGA or megapixel), because that would expand the required memory over a limit, which cannot fit to standard CMOS chip.

## *4.2 1D Processor Arrays*

The 1D processor arrangement (Fig. 8) is constructed of a linear processor array with local communication between the processors. The processors operate either in single instruction multiple data (SIMD) mode, or they are nonprogrammable special purpose ones. Each processor unit has a local memory. The executable operator types are defined from the aggregated memory size of the array rather than the individual memory size of the processors. The 1D processor array can be integrated with either a line sensor or a sensor array. The number of the processors can be either as many as the number of the pixels in an image line (fine-grain) or smaller (coarse-grain).

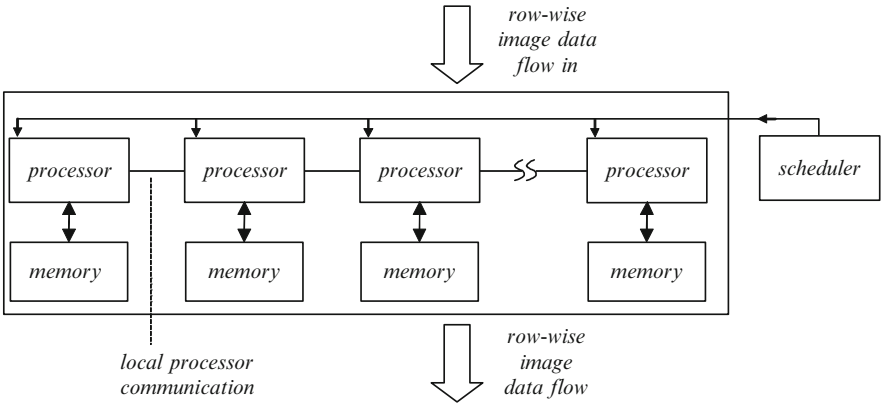


**Fig. 8** 1D processor arrangement

### 4.2.1 1D Processor Arrays with Line Processing Capabilities

A 1D processor array with line processing capabilities has enough aggregated memory to store a few lines. This efficiently supports the execution of *pixel-wise* and the *neighborhood* operators. Moreover, those *image* → *row/column* operators can be executed, where the propagation type is left to right or top to bottom. The *image* → *scalar* operators can be executed on this architecture also; however, the execution is more difficult, because each processor generates a subresult, which should be combined. For example, in case of seeking for the maximum pixel value, each processor finds the maximum in its column(s), and after that, the absolute maximum value should be selected in a second step.

### 4.2.2 1D Processor Arrays with Frame Processing Capabilities

The aggregated memory in this second type of 1D processor array is large enough to store entire frames. With this memory size, its capabilities become similar to a 2D coarse-grain processor architecture. It can execute practically the same operator set as the 1D with line processing capabilities, except it can calculate *image* → *row/column* operators in arbitrary direction.

## *4.3 2D Processor Arrays*

Two-dimensional processor arrays are applied either as a foveal array of a high resolution sensor, or as an embedded processor array next to a sensor array (Sect. 2). Since they can handle entire frames (or windows), we cannot distinguish them according to their aggregated memory sizes. Rather, we can separate them according to their processor density.

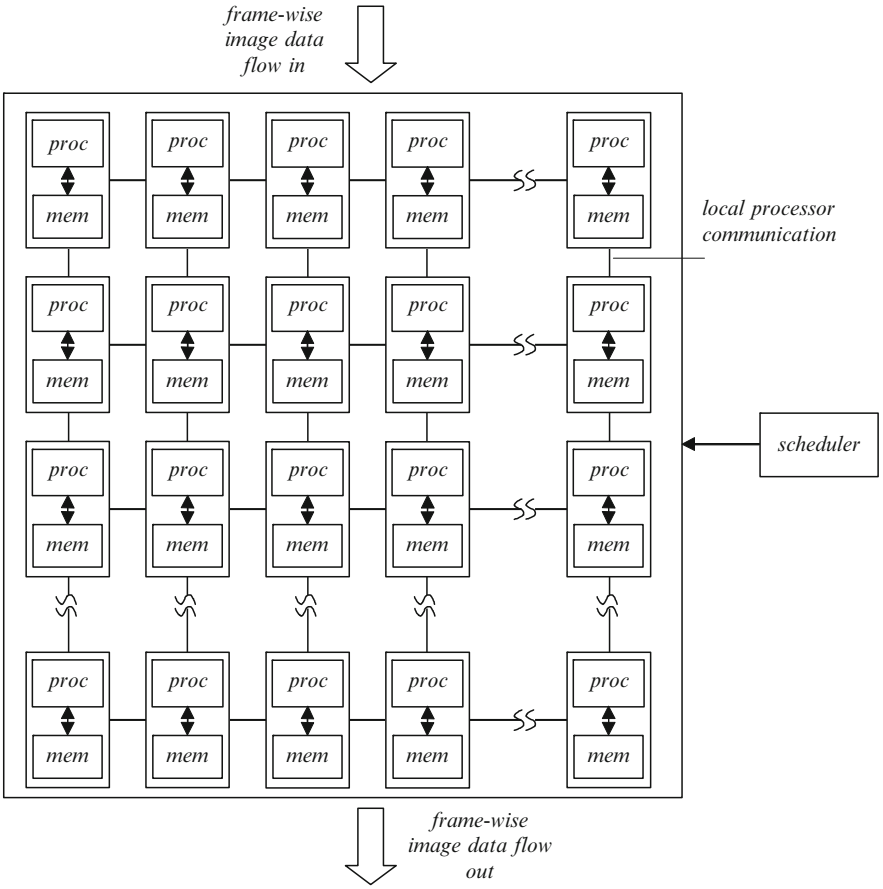


**Fig. 9** 2D processor arrangement

These processor arrays provide ultra-high processing capabilities. FPSP chips, equipped with this kind of engine, can easily reach above 10,000 FPS image capturing and evaluation (visual decision making) real time [9]. These processor arrays are SIMD architectures (Fig. 9).

### 4.3.1 Coarse-Grain 2D Processor Arrays

The coarse-grain 2D arrays can efficiently execute the *image* → *row/column* operators in all directions, the *pixel-wise* and the *neighborhood* operators. They can execute the *image* → *scalar(s)* operators also; however, the results need some post-processing as it was discussed in Sect. 4.2.1. These are typically digital processor arrays.

### 4.3.2 Fine-Grain 2D Processor Arrays

Similar to coarse-grain arrays, the fine-grain 2D arrays can efficiently execute the *image → row/column,* the *pixel-wise* and the *neighborhood* operators. They are not good at the *image → scalar(s)* operations in the conventional way; however, they can be very efficient when they use nonconventional approaches. They apply typically mixed-signal or bit-sliced digital processors.

The strength of the fine-grain processors is coming from the application of the nonconventional processing approach, because their mixed-signal and distributed asynchronous logic processor units can execute some *image → scalar(s)* and repetitive *neighborhood* operators with ultra-high speed. In these cases, the "let the physics do the computations" approach is used. The most important of these operators are:

- Global logic (AND/OR);
- Mean;
- Isotropic or anisotropic diffusion;
- Active pixel coordinate position readout;
- Object size estimation;
- Extremum (value and location);
- Repetitive binary morphologic operations (hole filling, recall, skeleton, centroid [15], etc).

The global logic is implemented in a way that a metal wire grid is set to weak Vcc voltage level through a resistor. In each node (processor cell), a transistor connects it to ground. The gate of the transistor is connected to the logic pixel value. Where it is high, the transistor opens. One open transistor is enough to pull the array down to zero. In this way, global OR is calculated.

By removing the pull-up transistor, and connecting a capacitor with the actual pixel value in each node, the same metal grid calculates the mean operator. After the transient decays (charge distribution is completed), the average of the pixel values will appear on each node.

Isotropic diffusion operator can be implemented on a resistive grid, by connecting a capacitance to it with the pixel value in each node. The strength of the diffusion (deviation) can be controlled by the transient time or by the resistance value. By controlling the resistance value locally, one can implement anisotropic diffusion. In case of nonlinear resistance, nonlinear diffusion can be implemented [16].

Coordinates of active pixels can be read out by scanning the pixels line-wise in an asynchronous way [2, 7]. From the introduced chips, ASPA (chapter *ASPA: Asynchronous–Synchronous Focal-Plane Sensor-Processor Chip*) is equipped with this capability.

Chips apply the operators above are described in chapters by Dudek and Laiho et al. (SCAMP-3, MIPA4k), and in [2, 6, 7] in the literature. The execution time of these operators is in the range of a few microseconds. This is 10–1000 times more power efficient than traditional digital solutions.

Extremum value and location can be identified by applying a comparator in each node. One input of the comparator receives a ramp and the other is connected to the

local pixel value. The output of the comparator is connected to a global OR network. The global logic network indicates when the ramp reaches first the maximum/minimum value in one of the node. Similar circuit for local maximum position identification is described in chpater by Zarándy et al. (VISCUBE chip).

Simple repetitive morphological operators (hole-filling, recall, etc) can be implemented either on the CNN [17] type chips [3, 4, 6, 7] in the analog domain, or by using asynchronous logic networks (chapter *ASPA: Asynchronous–Synchronous Focal-Plane Sensor-Processor Chip*, ASPA). While the execution speed of the formal one is one order of magnitude on the mentioned chips compared to a modern DSP, it is more than three orders of magnitude in the latter one [5]. This means that a grass fire type binary morphological operation can be calculated on a $128 \times 128$-sized lattice in 20 ns.

Complex morphological operators (skeleton, centroid [15]) can also be implemented on asynchronous logic networks with extremely power efficiency and ultra-high speed; however, they require larger silicon space [18].

### *4.4 Architecture Selection*

After analyzing the different architectures, a natural question arises: which one to use in certain application environment. The rule of thumb is that we need to apply:

- Embedded processor array, if high frame rate and low resolution are needed;
- Foveal processor array, if high frame rate and high resolution are needed;
- A sequence of single processors with medium size memories [5, 13, 14] in a pipeline arrangement, if high resolution and video speed are needed.

## 5 Conclusion

The anatomy of the different FPSP architectures is summarized in this chapter. This provides a help to easily navigate through the different architecture described in this book. More detailed architecture, operator, and processor structure analysis of the topographic devices can be found in [5].

## References

1. http://canesta.com
2. A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, A. García, C. Utrera, M. Dolores Pardo, J. Listan, 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, T. Roska, ISBN: 978–1–4419–1010–3, 2009

3. A. Paasio, A. Dawindzuk, K. Halonen, V. Porra, Minimum Size 0.5 Micron CMOS Programmable $48 \times 48$ CNN Test Chip European Conference on Circuit Theory and Design, Budapest, pp. 154–15, 1997
4. S. Espejo, R. Carmona, R. Domingúez-Castro, A. Rodrigúez-Vázquez, CNN Universal Chip in CMOS Technology, Int. J. Circ. Theor. Appl. 24, 93–111, 1996
5. A. Zarandy, Cs. Rekeczky, 2D Operators on Topographic and Non-Topographic Architectures Implementation, Efficiency Analysis, and Architecture Selection Methodology, Int. J. Circ. Theor. Appl. (CTA), Article first published online: 29 Apr 2010, DOI: 10.1002/cta.681
6. S. Espejo, R. Domínguez-Castro, G. Liñán, Á. Rodríguez-Vázquez, A $64 \times 64$ CNN Universal Chip with Analog and Digital I/O, In Proc. ICECS'98, pp. 203–206, Lisbon 1998
7. G. Liñan Cembrano, Á. Rodríguez-Vázquez, S. Espejo-Meana, R. Domínguez-Castro ACE16k: A $128 \times 128$ Focal Plane Analog Processor with Digital I/O Int. J. Neural Syst. 13(6) 427–434, 2003
8. P. Földesy, Á. Zarándy, Cs. Rekeczky, T. Roska Configurable 3D Integrated Focal-Plane Sensor-Processor Array Architecture, Int. J. Circ. Theor. Appl. (CTA), 573–588, 2008
9. Á. Zarándy, R. Domínguez-Castro, S. Espejo, Ultra-High Frame Rate Focal Plane Image Sensor and Processor, IEEE Sensor J 2(6) 559–565, 2002
10. R. Wagner, Á. Zarándy, T. Roska, Adaptive Perception with Locally-Adaptable Sensor Array, IEEE Trans Circ Syst. I, 51(5), 1014–1023, 2004
11. A. Zarandy, P. Földesy, T. Roska Per-Pixel Integration Time Controlled Image Sensor, ECCTD05 – European Conference on Circuit Theory and Design, Cork, Ireland, pp. III-149–III-152, August 2005
12. http://www.suss-microoptics.com/products/microlens.html
13. Z. Nagy, P. Szolgay Configurable Multi-Layer CNN-UM Emulator on FPGA. IEEE Trans. Circ. Syst. I: Fund. Theor. Appl. 50, 774–778, 2003
14. Cs. Rekeczky, J. Mallett, Á. Zarándy, Security Video Analitics on Xilinx Spartan – 3A DSP, Xcell J. 66, fourth quarter, 28–32, 2008
15. K. Karacs, Gy Cserey, Á. Zarándy, P. Szolgay, Cs. Rekeczky, L. Kék, V. Szabó, G. Pazienza, T. Roska, Software Library for Cellular wave Computing Engines in an era of kilo-processor chips, Version 3.1, Budapest, Cellular Sensory and Wave Computing Laboratory of the Computer and Automation Research Inst., Hungarian Academy of Sciences (MTA SZTAKI) and the Jedlik Laboratories of the Pázmány P. Catholic University, 2010
16. P. Perona, J. Malik, Scale-Space and Edge Detection Using Anisotropic Diffusion, IEEE Trans. Pattern Anal. Mach. Intell. 12(7), 629–639, 1990
17. L.O. Chua, T. Roska, Cellular Neural Networks and Visual Computing, Cambridge University Press, 2002
18. A. Lopich, P. Dudek, Architecture of Asynchronous Cellular Processor Array for Image Skeletonization, Circ. Theor. Des. 3, 81–84, 2005