Digital implementation of the cellular sensor-computers

Péter Földesy, Ákos Zarándy, Csaba Rekeczky, and Tamás Roska

Abstract

Two different kinds of cellular sensor-processor architectures are used nowadays in various applications. The first is the traditional sensor-processor architecture, where the sensor and the processor arrays are mapped into each other. The second is the foveal architecture, in which a small active fovea is navigating in a large sensor array. This second architecture is introduced and compared here. Both of these architectures can be implemented with analog and digital processor arrays. The efficiency of the different implementation types, depending on the used CMOS technology, is analyzed. It turned out, that the finer the technology is, the better to use digital implementation rather than analog.

Index Terms

Focal plane, sensor-processor, parallel processing, SIMD, fovea, CMOS sensor, CNN-UM

I. Introduction

Focal-plane cellular sensor-processor chips [6]-[8] traditionally used the combined sensor-processor array architecture. The common features of these architectures are that there is a one to one correspondence between the sensors and the processors, and the sensor and processor arrays are mapped into each other. Since the processors in these implementations occupy significant silicon area, the pitch size of these sensor-processor chips is between 50-200 microns. This yields a low fill-factor (5-15%); moreover, it prevents building high resolution (like 1024x1024) arrays. The largest operational combined sensor-processor array is the ACE16k [8]. It was implemented on 0.35 micron technology, and its size is 128x128. However, this focal-plane sensor-processor array can easily capture and process up to 50,000 FPS real-time, which makes them the fastest image processor devices in the world; it is not usable in many applications, where higher resolution is required. A new version of the ACE16k, currently under development, will step down to 0.18 micron, and reach roughly 20-micron pitch size and QZIF resolution.

Some applications, like surface inspection, image enhancement requires the analysis of the entire image continuously, hence the one-to-one correspondence of the sensors and the processors is needed.

Analogical and Neural Computing Laboratory, Hungarian Academy of Sciences, and the Jedlik Laboratories, Faculty of Information Technology, Pázmány Péter. Catholic University, Budapest; email: [foldesy,zarandy,rcsaba,roska]@sztaki.hu; web: lab.analogic.sztaki.hu

However, in other applications, like object tracking, or object identifications, navigation, etc, where smaller parts of the scene should be precisely analyzed only, an active fovea, navigating in a high-resolution image is more efficient. Though there is no single chip implementation of the foveal architecture today, a vision system, called Bi-i [9] was built based on this structure, and proves the viability of this architecture.

In the first part of this paper, we are going to introduce the digital implementation of the cellular sensor-computer architecture. In the third chapter, we describe the instruction set and computational performance. Later on, considerations of the technology scaling down and its effect on the implementation are given. Later on, implementation studies are presented and finally we conclude our work.

II. System Architecture Overview

The system architecture of the cellular sensor computer is introduced here has been developed to integrate the essential tasks in the same entity, namely the sensing, the tightly coupled parallel processing, and a top level communication and algorithmic control. The name of the game in all focal plane array processor implementation is the spatial resolution. The higher the spatial resolution (pixel count) the more precise image sensing and processing operations can be achieved. This certainly means that we have to minimize all the circuits, to be able to shrink the pitch size as much as possible.

In the digital domain, the simplest processor techniques are the bit sliced architectures. Though these processors can deal with a single bit in a clock cycle, in a sequence of clock cycles, it can calculate with arbitrary precision numbers. In image processing applications, 4-16 bits precision is typically enough. The advantage of the architecture is that the execution time is proportional with the precision, which makes possible finding trade-offs between precision and execution time.

From implementation point of view, the integration of the different functionalities, like sensing, processing, and control, means a solution for different preferences. These preferences can be high speed sensing, less design effort, or high spatial resolution. The different aims directs to different implementation of the same system idea. Our answers for the above three preferences are the integrated sensor-processor array, the separated arrays, and the foveal architecture (see Figure 1.). In the later chapters, we show other, finer tradeoffs found inside these basic classes.

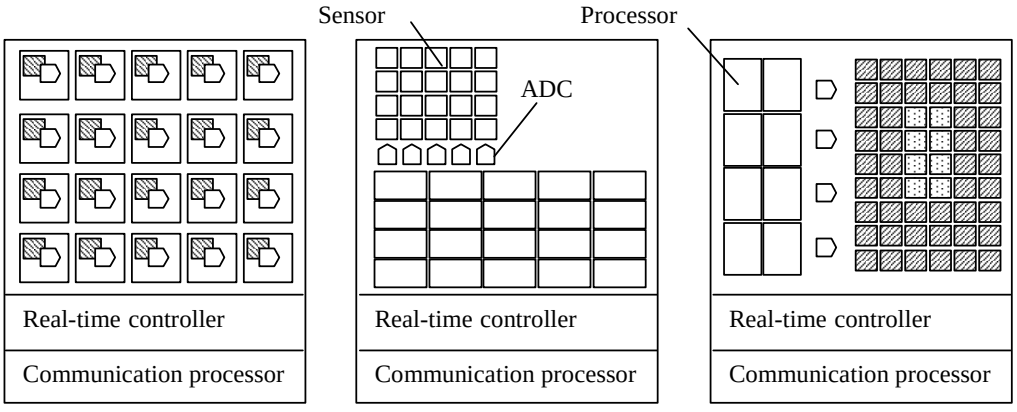


Figure 1. Three basic configurations of sensory and processing kernel integration.

A. Information flow

As in any complex systems, we can identify several levels of hierarchy in the cellular sensor computer architecture: sensor, processor, array, and system. The most interesting issue is that how the sensed scenes are processed and represented throughout this hierarchy.

The lowest level tasks, such as fixed-pattern-noise correction of the sensors, are performed by the individual cells. Than the information abstraction (e.g. feature extraction) goes upward in the hierarchy, from sensed raw maps to simple decisions affected by a small neighborhoods of cells. Meanwhile, high-level tasks, like the focus of attention or adaptation mechanisms of the sensing medium, are initiated from high level down to individual sensors. At this level, the need for maintaining a connection with a wider environment incl. other sensors, equipments, vehicle control also arise.

For low-level abstraction, a basic instruction set is need that is capable to support the elementary topographic processing operations and to perform local adaptation of the sensors. These requirements motivated the cell architecture: at the top level two units in charge for control and communication tasks. We show that these connections exist even in different physical forms of the architecture, such as the integrated or the separated sensor-processor arrays.

B. Sensor-Processor Cell

The workhorse of the architecture, the cells, can be summarized as a sensor-processor complex with local and global interconnection. As our primary aim to prove that the cell processors implemented in a digital logic is a feasible solution, we should address the question of interfacing the analog sensors to the digital circuitry. In contrast to analog implementation, the discrete time sampling and processing is must be used. Since, the sensors mostly produce analog signals reflecting the intensity of some physical phenomenon, an analog-to-digital converter is included into every cell to perform not only the time, but value discretization as well.

The converted signals are corrected (e.g. FPN reduction, gamma correction), processed (e.g. filtering, threshold, etc.), and stored in the cell processors. The reason for integrating sensors, converters, and a processor in a single entity, not only the accumulated sensing speed, but the unique possibility to control the sensing mechanism at the same rate. For this purpose, a local feedback path is inserted into the cell complex, coming from the processor kernel towards the sensor (or sensors) by means of digital-to-analog conversion. As we show later, this conversion could be as simple as pulse width modulation or a delayed deactivation moment.

In order to fulfill the topographical processing tasks (e.g. active wave propagation or morphological feature extraction), the processor kernel has been chosen similar to any general-purpose processor architecture: arithmetical resources, registers, local data storage, and external ports.

C. Cell Array Formation

The next architectural level is the integration many of such sensor-processor into a coherent array. We would like to emphasize, that the practical task of mass instantiation of cells is as difficult and complex as the cell design itself.

The topographic operations require spatially identical architecture and functionality (apart from local data dependency); hence equal cells are placed in a regular array driven by the same instruction flow.

For the shake of simplicity, the cell array grid is Cartesian and each cell is connected to its finite neighbors in horizontal, vertical, and diagonal directions in the same way as the CNN-UM architecture formed [1]-[3]. Beside the local interconnection, global sparse connectivity, and on the periphery, boundary cells are included in the array forming.

In a simple way, the digital portion of the array architecture is quite similar to of the modern FPGAs and other reconfigurable computing engines: reconfigurable resources, distributed memory blocks, programmable interconnections. The distinguishing feature of the cell array from this point of view is the shared, uniform configuration setting, and the single-cycle, run-time full reconfiguration capability. The price for the run-time reconfigurability is significant. However, at first sight the local and global data exchange seems to be dominant, it is not so. The increasing cell complexity and versatility among with the continuous reconfigurations implies one to two orders of magnitude higher data throughput between the central controller and cells, than in between the cells themselves.

D. Sensor-array and Processor-array architecture

By mixing the inherently analog sensing and digital processing results in a unique possibility for adaptive sensing, on the other hand rises an extreme design challenge.

The separated sensor and processor array gives a solution for splitting the two different functionalities. This separation has been done in a way, that the advances of integration remain, namely quick bidirectional connectivity between sensors and processor array is preserved. Apart from the one-to-one processor and sensor parity, the need for large sensory resolution with local fovea exists. To address this issue, we can break the symmetric correspondence and inflate the sensory array and associate time-by-time the processor resolution to different window of the sensed view.

E. Control and communication modules

The execution of the algorithm flow on the cellular sensor computer naturally raises also the need for continuous monitoring of the environment and the control of the cell array. These two rather different tasks can be satisfied best with two dedicated units, namely with a real-time processor and a communication processor. The real-time processor is in charge for quick reconfiguration of the cell array to perform operations as microcode sequences, and gathering or abstracting information coming from it. The communication processor is responsible for handling the uneven requests from the environment and from the array, provides standard protocol format for decisions, data, and images.

III. Cell Structure And Array Formation

The structure of the sensor-processor cell has been chosen to meet the requirements of the relatively huge number of circuit elements, control lines, and it still leads to compact solution. In this chapter, we give architectural details of the processor kernel structure, the sensory integration both in integrated and separated sensor and processor array architectures.

A. Cell Structure

Most of the complex topographic and sensory signal processing operations can be divided into smaller functions, such as series of conditional branches, additions, or comparisons. These basic functions are also composed of a set of single bit addition and data manipulation steps (atomic steps). By selecting common items in the different atomic steps, a highly compact architecture can be build.

Such a structure enables high flexibility and throughput rate by reconfiguring groups of basic units for different tasks.

For mapping these ideas to architecture, a versatile structure of a cluster of blocks and flexible crossbar switches in between has been chosen (similar to FPGAs). The choice for a crossbar switch for intra-cell interconnectivity serves the parallel data access for the multi-input blocks and on-the-fly function mapping. The cell components are the followings (see Figure 2.):

- Processor elements
  - Reconfigurable arithmetical units
  - Temporary register file
  - Digital memory
  - Crossbar switch including condition flags
- Communication elements
  - Global and local area interconnection ports
- Sensory block
  - Sensors and their signal conditioning units
  - AD and DA converters

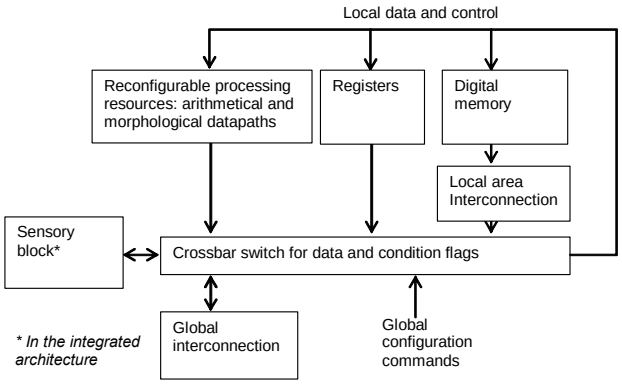


Figure 2. The cell structure of the sensor-processor architecture.

B. Processor Elements

In the selection of processing kernel, the main challenge is to reach small cell size, or better say a high performance/area ratio. A natural way for compactness is the specialization. For the appropriate architectural choice, the proportion of the specialization versus functional versatility must be selected taking care of other design issues such as control and data throughput rate. For this choice, one should consider all levels from functionality to circuit design.

In a wider scene, one can find many ways towards mass processing or parallel processing. The main driving force is the technology scaling down, which enables the integration of increasing number of processing engines. One can find excellent design space explorations [17] for such trade-offs from the basic 1-bit processor up to the array of floating-point units. From they results, it can be concluded, that for a given functional complexity, the simplest architecture with the highest number of cloned processors provides the best performance per unit area (see also [18]).

The near-sensor topographic processing inserts another constraint into the design, namely the existence of the regular grid of sensors. The processing elements and other components should be

spread around this regular grid. This constraint limits not only the complexity and array size of the processing units, but the number of maximally routable control signals also. Furthermore, the rectangular grid in the processor array generates a very complex layout arrangement problem, which requires very large efforts from the design team to solve.

Based on the above considerations, we have selected a bit-serial architecture for the processor elements. It holds for any element in the cell architecture, the registers, the memory, as well as the communication lines. The processor kernel is a simple unit, holding the main features of a general-purpose processor. It has bit-serial processing resources, registers, memory access, condition flags, and a dataflow (i.e. a crossbar switch) between the elements. In the next sections we overview the content of these blocks.

1) Processing resources

The processing resources are as simple elements as a collection of a 4-input LUT, SR flip-flop, a full-adder, and some other simple gates (see Figure 3.). These are combined later by the crossbar switch to form complex computations (i.e. they are like configurable logic blocks, CLBs, in an FPGA).

The reason for the double datapaths in this block is double fold. The full-adder path is clearly for addition/subtraction-based arithmetics, while the LUT/SR flip-flop path is for accumulation of series of bit-wise logic operations. The second goal for the doubled datapath is the capability of parallel extraction of information regarding to saturation condition, 2's complement and unsigned arithmetics, local addition/subtraction selection, and comparison type operation acceleration.

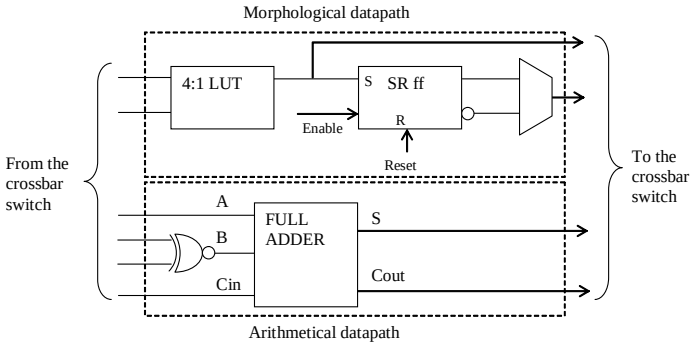


Figure 3. The reconfigurable logic block.

2) Storage and local interconnection

The local data storage and temporary registers together are crucial for high performance, simply by providing information for the processing resources in time. Both storage units are organized into bit-serial random access 1-D array with locally settable write enable and a local input source selector.

The local intra-cell interconnection is placed near to the main memory data access. As can be seen in Figure 4., the different blocks of the cell can access the memory through a neighborhood selector. This solution merges the neighborhood connectivity into the same environment together the memory or registers. In practice, all the cells can work with the data stored in the neighbors, as it was their self-memory content. This solution significantly reduces the application development time by simplifying the neighborhood operators implementation and providing a more conventional programming model.

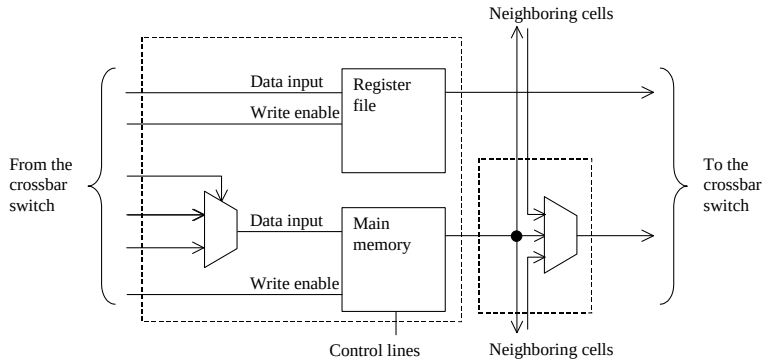


Figure 4. The storage units and the local area interconnection.

3) Crossbar switch

The crossbar switch is responsible for quick reconfiguration of the resources by different connections, and for temporary storage of the input/output values of the different units. The former task is implemented as a sparse connection matrix. The available connection paths are selected by profiling a large collection of operations. The storage and the synchronization of the unit outputs to specific module inputs are achieved by enabled edge-triggered flip-flops and multiplexers at each connected point in the switch matrix (see Figure 5). Hence, the reconfiguration instructions are collections of multiplexer control and clock enable signals of this matrix.

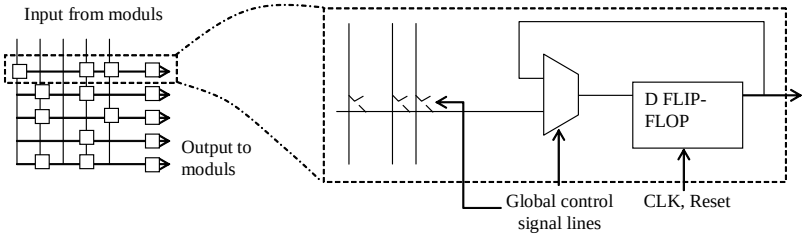


Figure 5. Example for the sparse connectivity matrix, and one element of the crossbar switch.

4) Global interconnection

The global interconnection is prepared for three different data transfer modes, namely to-from selected cells or row/column of cells, for single data value transfer for all the cells, and for single bit extraction from the whole array by a multi-input logic gate.

For data transfer, the straightforward solution is to chain all the cell memory blocks into a large memory array, augmenting it by some address, data, and control lines. The resulting architecture will behave exactly like a standard memory.

The existence check of black (or white) pixel in the image or in a portion of it could highly increase the efficiency of the binary mathematical morphology. A cell wise input logic “OR” gate is used for this single value extraction.

5) Data condition flags

The local data dependent execution is elementary for complex operations in any SIMD architecture. For this purpose, we allocated some flags mapped into the crossbar switch that enables or disables the memory/register update. Through the programming of the enabling flags, not only the algorithm flow can be manipulated, but the common task of region-of-interest (ROI) processing is solved as well.

C. Controlled sensing

The outstanding feature of integrated sensing and digital processing can be explored fully if we understand the combined nature of the continuous time dynamics of the measured physical phenomenon and the discrete time processing.

At the lowest level, the sensor dynamics need to be captured. Since the attached controller (processor) is digital, our model should be discretized to yield a proper control system. At this control level, the system become time based, as the events are triggered by given times. At each measurement time instant, an observation is provided by sensor reading to the processors, which will in turn produce an input to them via an actuator. This low-level control loop inherently requires the bidirectional conversion between the two different portions. On higher level, the algorithmic background will interpret the sensed data. The reader is redirected for theoretical background and the biologically motivated utilization of the integrated sensing and processing in [2]-[3].

For the shake of simplicity, let us consider the case of visible light sensing by a simple linear sensing medium sampled by temporal integration. This case covers the commonly used CMOS active pixel sensors or CCD sensors. In this scene, the sensory signal is an integrated value of the light intensity over a discrete period and the actuation is the discrete control of the integration. The integration in practical circuits must start from an initial value by a reset event; its control is governed by the initial value, the time instant of the reset event, and the sampling period (see Figure 6).

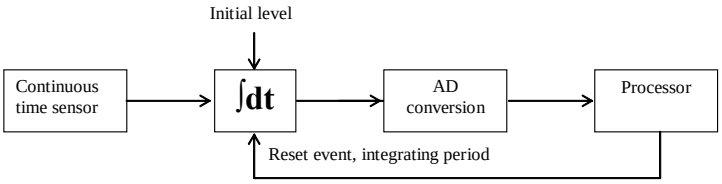


Figure 6. Sensor control model of the architecture.

Numerous advances can be gain from this approach. Several basic results of digital signal processing and still camera technology could be applied. The representative examples are the bad pixel or fixed-pattern noise removal, gamma correction, oversampling or interpolation for reducing the temporal noise and increase the sensed dynamic range [5], [10]-[12], [29], or motion detection during the integration time.

In the rest of this chapter, we describe the conversion between the sensor and processor by showing a solution for the described case. Later on, some interesting possibilities of sensing are listed.

1) Sensor signal A-D conversion

The conversion of the sensory output has been assimilated to the bit-serial architecture of the processor. For both circuit area and simplicity, single-slope type AD conversion has been chosen. In architecture, it is composed of per-cell comparator and a global analog signal source. During operation, a simple algorithm stores data in the memory of the processors depending on the comparator output.

The basic idea of the distributed single-slope ADC is to generate a continuously increasing digital value in conjunction with a ramp signal provided by the global analog source. The integrated signal of the sensor is compared to the analog ramp signal, and whenever their output becomes equal to the ramp signal, the associated digital value is stored. Hence, after sweeping through the complete dynamic range, every sensory output will have a digital representation, and the AD conversion is done [29] (see

Figure 8). In order to increase the flexibility of the architecture and to separate the conversion time requirement from the integration period, the converter is fed by a sample-and-hold unit (Figure 7).

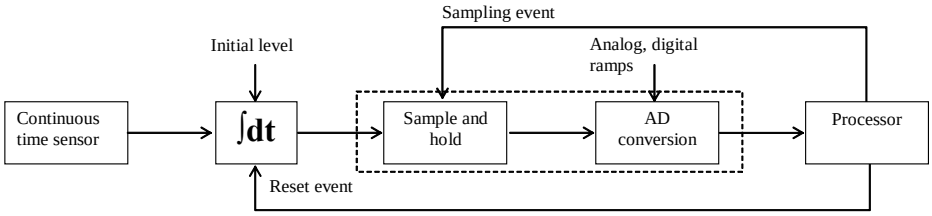


Figure 7. Sensor control model of the integrated sensor-processor architecture.

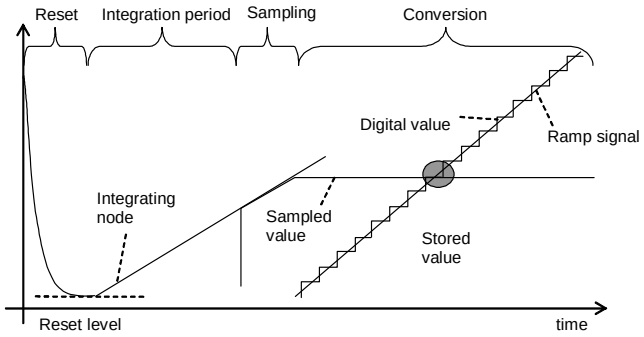


Figure 8. Reset event, integration, sampling, and conversion periods, and the working of the single-slope AD conversion.

2) Sensing schemes

Given by this flexible architecture, one becomes capable to impellent existing or novel non-traditional sensing methods. As we allocate the different events of the control loop into local control, the more interesting possibilities opens: By consecutive snapshots, there is the possibility to estimate more precisely the intensity map of the visual scene [4]-[5] by weighted averaging the samples. The extreme differences in a view result in under-exposure and saturated regions. By adaptation of algorithms of [19], [20], one can feed back this information to the sensing medium, now, through the integration time, achieving a realistic, compressed dynamic range image. In the described architecture, not only the inter-frame adaptation can be solved, but using the intermediate samples, an intra-frame adaptation too. Furthermore, the motion blur free captures can be produced exactly in the same way as [24] describes.

D. Separated sensor and processor array

As a distinguishing feature of the integrated sensor-processor architecture is the continuous sensory control loop, in the physical separation of the sensing and processing, we seek for a solution to preserve this property. After considering different possibilities, the architecture has been found to preserve this feature have three arrays, namely

- a sensor array of resolution, with per-pixel sampling information,
- a 1-D AD converter bank,
- and the processor array,

where, in principal, the sensor array resolution is identical to of the processor array (see Figure 9.)

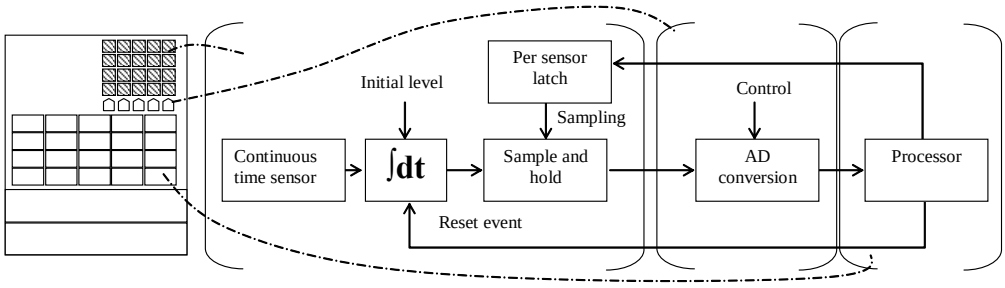


Figure 9. The separated sensor and processor array data loop.

At first sight, the column-wise digitalization seems to be slower than the distributed single-slope conversion, but it is surprisingly not so. The different implementations confirm that the conversion rate remains in the same range for various array sizes [31]-[33]. From the other hand, the control datapath from the processors towards the sensor array shows up two differences from the integrated case: the transfer rate that limits the temporal resolution of sampling/integrating period, and the synchronicity of the control throughout the whole array. The first issue can be addressed by high speed, column-wise bus system, while the second one can be guaranteed by master-slave latch type in the sensor array. It turned out, that the temporal resolution of 5-10 microsecond is enough for high quality adaptation mechanisms [19]. This rate can be maintained with no special requirements on the data bus systems (e.g. 100 MHz) up to reasonable array sizes.

E. Foveal architecture

In many visual application fields, large image sensing cannot be compensated by a smaller one with embedded processing capabilities. On the other hand, the foveal approach for navigation with high efficiency in a large view by a smaller fovea window is a satisfying compromise [35]-[36]. To considering this field, it is worth to note that the sensory and processor resolution can be also separated by simple multiplexing at the communication channels as an extension of the separated architecture.

In order to satisfy the higher sensory resolution needs, the sensory backward loop can be cut for further increase the sensory resolution [34]. The resulting architecture is composed of a classic sensor array (e.g. a simple active pixel type), column-wise converters, and a processor cell array. With this reasoning, we arrived to the third basic architecture type of cellular sensor computer.

IV. INSTRUCTION SET AND OPERATION SPEED

In this chapter, we overview some distinguishing features of the digital processor as a continuously reconfigured bit-serial engine, and we draw performance figures for it. For the description of the architecture's working mechanism, we overview the classic bit-serial operations with focus on resource reusability.

A. Bit-serial Implementation

Several work concluded that the computational throughput per unit area is higher for bit-serial arithmetics than bit-parallel (e.g. [18]). It is due to isomorphic data mapping, straightforward pipeline structures, and relatively high-speed logic. The reason why we cannot found several bit-serial processors around is that, these architectures solves usually very well defined tasks with hard wired architecture [13]. We show that the combination of the compact architecture and the flexibility of run-

time reconfigurability, a versatile but small computing engine can be build. The work in [21] describes FPGA architecture for bit-serial datapath synthesis. Although, it is a combination of the reconfigurability and bit-serial arithmetics, similar to our case, they do not consider run-time reprogramability.

The operations are split into bit level logic operations due to the bit-serial architecture. During operation, the data flow can be imagined as combination of data and control streams. The data bit streams come from different sources (e.g. memory or registers) they are combined in the arithmetical units and the resulting streams or single bits are stored in one of the sinks (e.g. the memory). The units' switches and the crossbar switch control the data stream flow in a form of continuous reconfiguring control stream.

The bit-serial arithmetics [13]-[14] in general can be described by its basic resources, that are unit-delay, logic inversion, two-input gates, full-adder, and by their connectivity. Let us now recall the basic operations and how they are mapped to the cell architecture.

1) Addition, subtraction, comparisons

Supposing the architecture showed in Figure 10., the two operands are fed to it starting from the least significant bit up to the most significant one. Depending on the operation, the operand B is negated, and setting the carry bit to logic one at the first step, and subtraction is performed. The carry bit is mapped to a flipflop in the crossbar switch.

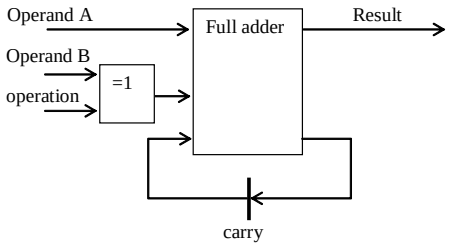


Figure 10. The basic scheme of addition/subtraction in the bit-serial processor.

The result is a bit stream containing the sum or difference. There is a problems exists, namely, the word length of the result is larger than the operands. As usually the word length is required to be constant, saturation handling is implemented to overcome this problem. When the result is larger or smaller than the available range, the resulting value is changed to the largest positive/negative values, respectively. For this solution, the existence of overflow/underflow must be detected first. The mathematical form of such a case is that the last carry value is different from the most significant bit of the result.

Regarding to the comparisons, the basic cases are the equivalence, greater and equivalent, and greater checking between the two operands. To perform this checking, we can use the other datapath of the processor to monitor and collect the resulting bitstream. For example, in the equivalence checking (see Figure 11.), after resetting the SR flip-flop and setting the carry bit, the B operand is subtracted from the A operand, while the morphological datapath will issue logic one only if there was not a single bit difference in the subtraction result (i.e. operand A is equal to operand B).

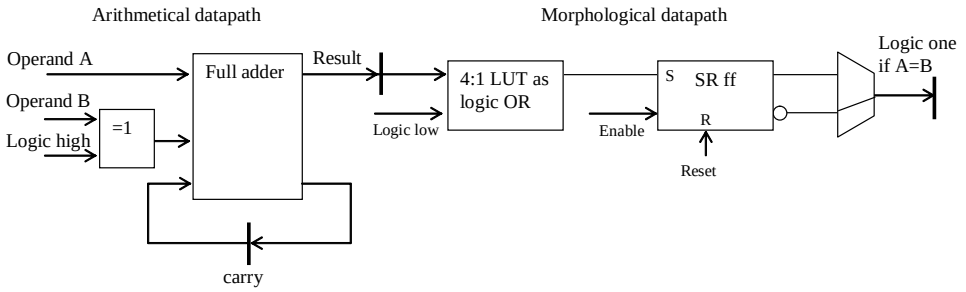


Figure 11. The two datapaths are configured for equivalence checking.

2) Multiplication

The multiplication is done in a serial-serial manner, that is both operands come bit-by-bit. In a general case, the serial-serial multiplication has a time complexity of $O(N^2)$, if the word length is N. On the other hand, in many grayscale low-level image processing tasks (e.g. convolution), the multiplicand is constant, hence a number of zeros can be found in them. Skipping the multiplication of the local operand is for these bit positions; the required time can be significantly reduced (e.g. for Gaussian kernels the ratio of ones in the neighborhood weights is about 20-30% after normalization to 8-bit).

Regarding to this solution, the local memory and register architecture is worth to mention. They have not only bit-serial interfaces, but also random access. Thus, the required shift operation in the bit-serial multiplication is done simply by modifying the addressing of them and skipping the unnecessary multiply-by-zero cases. There are some other tasks to be done as well: sign extension, truncation, and saturation handling. The sign extension is performed during operand reading by holding first the least and than the most significant bits in the operands' flipflops in the crossbar switch. The morphological datapath is used for overflow/underflow checking in parallel by monitoring the leading bits. The overflow/underflow detection result is handled during output storage e.g. in the memory by generating the proper saturating value and controlling the memory source selector. Note that how the datapath logics and the crossbar switch is used in different configurations during a single operation.

3) Morphological Operations

The binary mathematical morphology is a powerful tool for binary (black-and-white) image processing [15]-[16]. The most representative examples are the dilation, erosion, or ones that are more complex, such hit-and-miss or object logic operations.

The atomic step of the mathematical morphology is a logic sum, multiplication, or mixture of them in a finite neighborhood of the pixels described by the so-called structuring elements. By forming a bit-stream packet from all the neighboring pixels in the cells, the morphological datapath unit can easily used to perform any operation on them.

Let us describe the example of a hit-and-miss operator [15]. These operators are morphological equivalents of pattern matching, a well-known technique for matching patterns based upon cross-correlation. The task is the following:

$$HitAndMiss\,(A,H,M) = E(A,H) \cap E^c(A^c,M),$$

$$E(A,B) = \bigcap_{\beta \in B} (A - \beta),$$

where $E()$ is the operator of erosion, H and M is the hit and miss structuring elements and A is the binary input image, respectively. Streams can be formed from the neighboring values and the structuring elements. The neighboring values comes from the cell and the neighboring cells' memory. The structuring element stream is mapped to the 4-input LUT to invert or not the memory data stream (if the current position is part of the hit structuring element, the data is inverted, otherwise not), and finally, the SR flipflop is used for the accumulation of the formed data stream (see Figure 12.). The handling of the don't care positions in the structuring elements is performed by controlling the enable signal of the SR flipflop. At the end of the process, the SR flipflop contains the result of the operation.

The operations build up from the pattern matching, such as thinning, use this basic step and the consecutive logic addition or subtraction. It is worth to mention, that the propagative operations (such as thinning), which has a well defined steady state, can be monitored for such conditions by the global logic gate and stop further iterations.

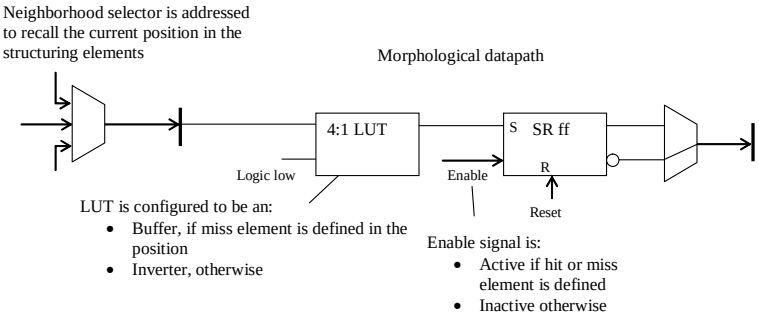


Figure 12. The processor is configured for mathematical morphology pattern matching.

B. Performance

In this chapter, we conclude some important operations and their execution time in clock cycles. In general, the execution time is closely related to the number of involved bits, with the addition of 2-3 cycles latency due to the pipelined operation caused by the crossbar switch.

Let us first summarize the basic operations, and later on more complex operations come that are composed of the basic ones in Table I, and Table II, respectively.

Table I. Basic operation types

Operation name	Required cycles	Latency
Bit-wise logic operation	1	2
N-bit signed/unsigned addition	N	3
N-bit*2^K signed/unsigned accumulation in M-bit accumulator	M-K	3
Saturation handling, in addition to accumulation for N-bit output	N	2
Constant setting for n-bits	N	2
Comparison of one or two N-bit values	N	3

Table II. Complex operation types[#]

Operation name	Required cycles
8-neighborhood morphology I: dilation, erosion	11

8-neighborhood morphology II: iteration of a hit-and-miss operation with 8 sub-kernels (skeleton, thickening)	70
Threshold (grayscale to 1-bit)	12
Compare images	12
Image addition with saturation	18
Multiplication with constant (sustained)	98
General 3x3 convolution (sustained)	890
Sobel edge operator	134
Median filter in 3x3 neighborhood	630
Blurring in 5x5 neighborhood by Gaussian kernel of sigma=1, sum of weights are normalized to 255	675

[#] Grayscale images have 8-bit resolution.

V. Scaling Down And General Considerations

The circuit feature size scaling down effects the leakage power, process variability, reduces the supply voltages, the design productivity, causes signal integrity (SI) problems. From circuit design point of view, a sensor-processor system is naturally degraded by these difficulties too. As it is being a complex mixed-signal architecture, not only the relatively easily handled digital circuit performance degradation occurs, but also significant problems rise at the analog portion of the system [23]. These prospective difficulties can be overcome by the separated sensor and processor arrays.

In this chapter, we give an overview on how the scaling down affects the sensor-processor integration both in area and performance, and what solutions has been found for further advances, and we evaluate several tradeoffs.

A. Digital circuitry

For the speed and area estimations, we approximate the equivalent gate size of the processors, the area requirement of the memories, and later on, we calculate with general technology properties.

1) Logic complexity

As the processor kernel of the cells is quite simple and compact, we can easily count their gate size (see Table III). The size of an equivalent gate size in technology feature size depends on many things, but let us estimated it as being 300 λ^2. Hence, the processing kernel's size is about 33,000 λ^2.

Table III. Processor gate count

Unit name	Gate size
Processing resources	16
Local and global connectivity	10
Crossbar switch	84
Sum	110

2) Data storage

One of the most challenging issues is the local data storage. In high performance processors, usually they fill the silicon dies up to the manufacturable size with embedded memories, caches. The reason is simple, the intra-chip communication can be much faster than the off-chip memory access, resulting in

higher performance. In our case, the parallel data access is also a key feature; hence, the distributed per-cell memory structure cannot be avoided to fully explore the performance of the parallel architecture. This implies a complete memory block for every cell, including data bit array, decoders, read-write, and refresh mechanisms.

In general, one can choose different types of data storage in standard CMOS technology: dynamic or static memories, binary or multilevel (that is typical in flash memories). The dynamic and multi-level types are more compact, while the static and binary ones are more robust and do not require special data refresh. Furthermore, as any type of memory needs the common circuitry (e.g. decoders) the smaller array size must be favored to reduce the overall area, but with respect to the area of the supporting circuitry.

The selection of the proper memory type is affected by its robustness. In a relatively complex array processor, the aggregated amount of the distributed memory could easily exceed some megabits. In this order of magnitude, the yield of a less robust architecture is below the acceptable level without hard and soft error correction methods [22]. By the implementation of such a method the area advance of a compact solution rapidly vanishes due to the small memory blocks.

A good compromise has been found by the 3T DRAM type. This type is definitely more compact than the any SRAM, while it does not affected by the charge redistribution during readout. We can estimate the size of a bit 150 λ^2, while the size of an N^2 bit array including dynamic row and column selectors and the a simple precharge-evaluate readout scheme to be approximately 150 N^2 + 3000 N λ^2, where λ is the general technology feature size.

B. Sensing medium

With respect to optical sensing, the optical responsibility of the CMOS technologies reaches the end of road beyond the 0.18-0.13 micron feature size technologies [25]-[26]. This fact is due to many reasons, the increasing number of metal layers and the dielectrics between them, the aggressive silicon doping, high leakage current, the obscure low-K dielectric results in extremely low Quantum efficiency.

The straightforward solution is the separation of sensing from the processing technology in parallel with thigh coupling of the two entities. Although, several on-chip-surface, vertical sensor manufacturing techniques exists (e.g. amorphous silicon grow or 3D stacked chip architectures), they are not common and are mostly in experimental phase [27]-[28].

For easier estimates let us suppose the following dimensions for the integrated sensor-processor architecture cases, 5x5 micron sensor area and about 1000 λ^2 analog circuitry including comparator and signal conditioning, and 7x7 micron complete size for the separated one, and 5x5 micron classic active pixel sensor size for the foveal architecture, respectively.

C. Separated sensor and processor array

The separation solves the unique problem of tremendous mixed-signal circuit integration. For both arrays, the most appropriate method or even technology node can be used. For the digital portion, we can rely the above estimates and the complete separated sensor size. The size of the interconnection (i.e. converters) between the array, supposing a compact and pitch matched structure, let be about 1000 –2000 λ height.

D. Area/Speed figures

Now, we can sum up the portions and estimate the required area for the architecture variants. In the integrated sensor-processor architecture, the total cell area is estimated as (33,000 + 150 N^2 + 3000 N + 1000) λ^2 + 25 square microns, where N is the memory size of the cells and λ is the feature size. In the separated processor array, the kernel size is (33,000 + 150 N^2 + 3000 N) λ^2 square microns. The memory capacity for a cell, based on algorithmic requirements, could be limited to eight bytes (N^2=64). The ratio of the different component can be seen in Figure 13.

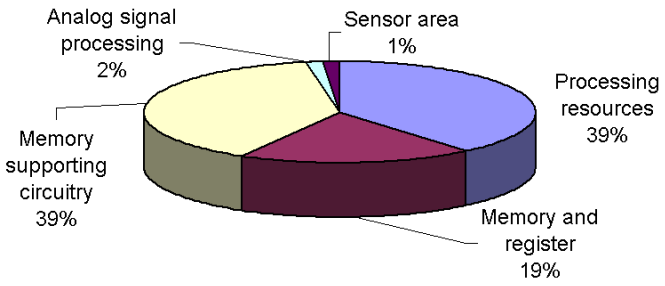


Figure 13. The ratio of different units in of the sensor-processor architecture.

Although, this estimates does not count with several properties of the technology scaling down (such as feature size scales more aggressively for gate width than the other dimensions, more metal layers available), gives an initial idea of the reachable complexity. For easier comparison, let us suppose a 1x1 cm square area for the cell array.

For the speed estimates, we must restrict ourselves to the limit of the global data throughput. As we mentioned, the global control mechanism of continuous reconfiguration requires dozens higher throughput rate, than the local interconnection does. Hence, the overall speed estimates cannot rely on the local clock speed, but on the across-chip one. A first-order estimates for the maximal clock rate can be derived from a simple time constant calculation:

$$t_{wire} \sim \rho\kappa\left(\frac{L}{\lambda}\right)^2,$$

where ρ and κ are the resistance and capacitance per unit area, L is the wire length, λ is the metal pitch. For the more realistic estimates, let us suppose the followings: lower 3-4 metal layers are used for local wiring and the increasing number of higher metal layers for global routing, 20% clock skew, L= 1 cm, low-K dielectric and Cu metallization for technologies below 0.18 micron, we get a rough clock speed limits as shown in Table IV.

1) Integrated sensor-processor architecture

The straightforward calculations show the array size and performance results for the integrated architecture in Table IV. Note, how the shrinking wire pitch results in lower operating frequency, balancing the increasing array size. For comparison of this performance, the most advanced available Texas Instruments DSP (the TMS320C6x series at 1 GHz clock speed) performs 4 GMAC@8-bit peak performance [37], a survey can be found on specialized engines in [30].

Table IV. Area estimates for the sensor-processor architecture and 1 cm^2 area

Technology	Cell size [micron]	Size of array	Clock [MHz]	Peak GMAC@8-bit
0.09	25*25	400*400	50-100	80-160

0.13	35*35	285*285	90-120	75-100
0.18	48*48	208*208	100-150	44-66
0.25	66*66	152*152	~200*	45

In the exploration of the finer architectural possibilities, we can modify the priority between sensor and processor resolution by share some resources between the abutted cells. As the sensory grid is a strict constraint in the physical layout design, only a few possibilities have reality. One representative example for the resource sharing is the increasing number of sensory block controlled by a single processor kernel. Although, this option can significantly extend the sensory resolution, it makes the practical design more challenging. Now, the sensory area is increased in parallel with the stored information within a cell. The changing relative performance and resolution ratio can be seen in Figure 14. as a function of the number of sensors per cell.

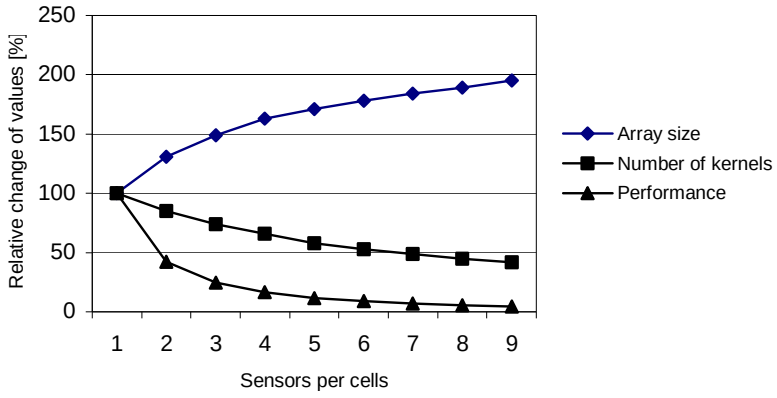


Figure 14. Trade-off between sensory array resolution and processing performance for unit area.

2) Separated sensor and processor array architecture

The separated architecture provides several additional solutions that can be applied, which was not available in the sensor grid restricted mixed-signal structure. These possibilities cover the hierarchical buffering of control lines, increasing the pipeline depth and operation frequency, increasingly the resource sharing, in general, the whole arsenal of the digital design flows.

The most relative area consuming component is the memory addressing logic that can be easily shared by even more than one processor. In this context, we can calculate new optimums for different resolution preferences simply by sharing the pixels (i.e. memory) by processor kernels in a similar way then we did in the integrated case. The Figure 15. shows the changing relative properties of processor arrays with different number of handled pixels per processor. It is worth to compare this case to the similar evaluation of the integrated architecture. The separated architecture provides better array size and performance scaling.

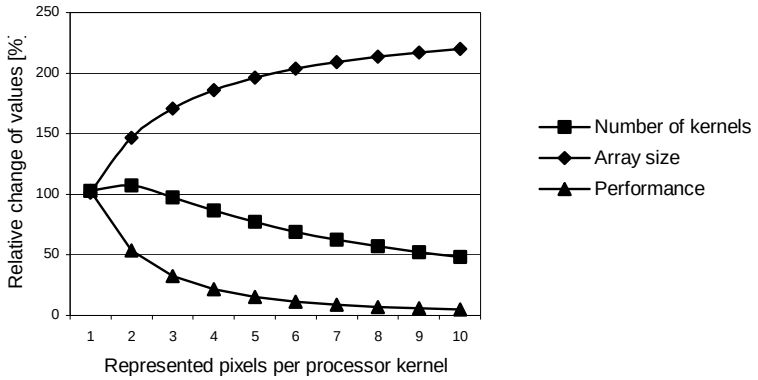


Figure 15. Trade-off between performance and the on-chip handled image size for unit area.

E. Foveal architecture

In the foveal architecture, as we stated, the sensory resolution is the selected priority with significantly larger resolution than the processor array has. For this architecture, the most interesting tradeoff is between the sensory resolution and the window/processor array ratio for unit area (see Figure 16.). The message of this result is that a two-megapixel sensor can be integrated with a 128x128 navigating processor array into a 1 cm^2 silicon area using a mainstream 0.18 micron technology.

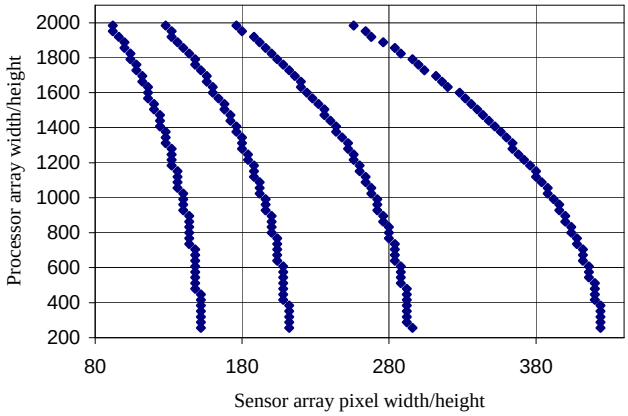


Figure 16. Trade-off between sensor and processor array size for different technologies (0.09, 0.13, 0.18, 0.25 micron from left to right side, respectively).

F. Yield considerations

In general, with the scaling down of circuit dimensions, the manufacturing reliability, the operating conditions dramatically worsen. With respect to the digital portion, the circuit reliability can be mitigated by several techniques, such as robust circuit solutions, built in element redundancy and attached self-repairing mechanisms (BISR), with a common term, by design for manufacturability.

The most challenging issue in the robust sensor-processor implementation is the preservation of the sensing uniformity and resolution despite of any circuit redundancy and BISR solution.

For the separated architecture, as the two entity (sensor and processor array) are separated, there is no restrictions to hold them in the silicon die. By advancing the digital portion with the manufacturing

trend, and holding the sensory system in a conservative technology, the whole system can yield a lot from the corresponding design methods having more feasibility.

G. Conclusion

In this chapter, we have calculated some important estimates about the possible implementations of the architectures in the mainstream technologies. We showed also, that the sensor-processor integration interfere with the aggressive scaling down trend in the sense that the classic light sensing capability of CMOS technology vanishes below 0.13 micron feature size. As an escape route, we analyzed the separated processor-sensor array architecture giving guidelines for finer tradeoffs.

VI. IMPLEMENTATION STUDIES

For the verification of the analytical results, we have designed several variants of both architectures at the 0.18 technology node (1P6M). In the examples, the goal was to satisfy some basic requirements: regularity for abutted cell array formation, correct operation in a large array, and complete functionality. These requirements have been verified, by backannotated parasitic simulations, some manufactured samples, and various analyses.

The first example is for the integrated sensor-processor architecture. In this example, we have selected that variant, where a processor kernel works with four identical sensory blocks. The sensory block composed of n+/p-substrate photodiodes, followers and other signal conditioning circuitry, and a comparator shared in time by the four sensors during AD conversion. The design effort was high, due to two reasons: the high number of control lines are laid into the gaps within the sensors, resulting in a significant crosstalk in between, furthermore, the signal routing within the cells was also a non-trivial task again caused by the sensor pitch matching. The resulting cell area is 60*60 micron square (see Figure 17.), pretty well in accordance to the estimations (62*62 micron). The targeted and checked operating frequency was 100 MHz.

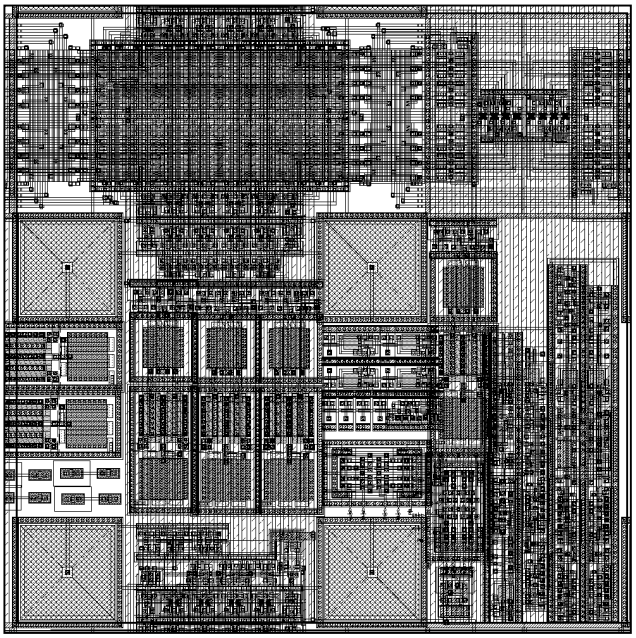
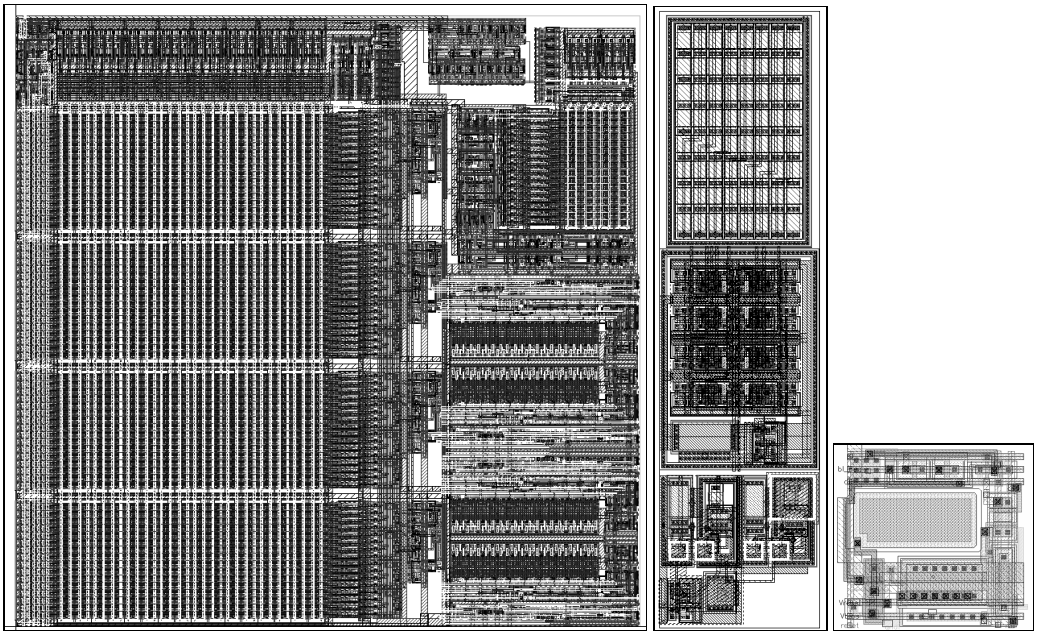


Figure 17. Integrated sensors and processor cell. Four sensors are shared by one processor. The size is 60*60 square micron.

The second example is an implementation of the separated sensor-processor array architecture. The processor kernel has been designed with again strong resource sharing: four pixel data has been denoted to a single processor, while blocks of four such processors has been grouped to share the memory decoders. The other two elements of the architecture are the pitchmatched AD converters and the locally controlled sensors. The AD converter is composed of a multilevel (5-bit) charge redistribution DAC with 8-bit final resolution and a successive approximation register. The conversion rate is a moderate 2 Mpixels/sec for each converter. The sensor is a n+/p-substrate photodiodes active pixel sensor with electronic shutter, that is controlled by a per-pixel latch (implementation details could be found in [20]). The Figure 18. shows the layout of the modules. In this example, the AD pitch is double of the sensory pitch, resulting in two instances of converters per column.

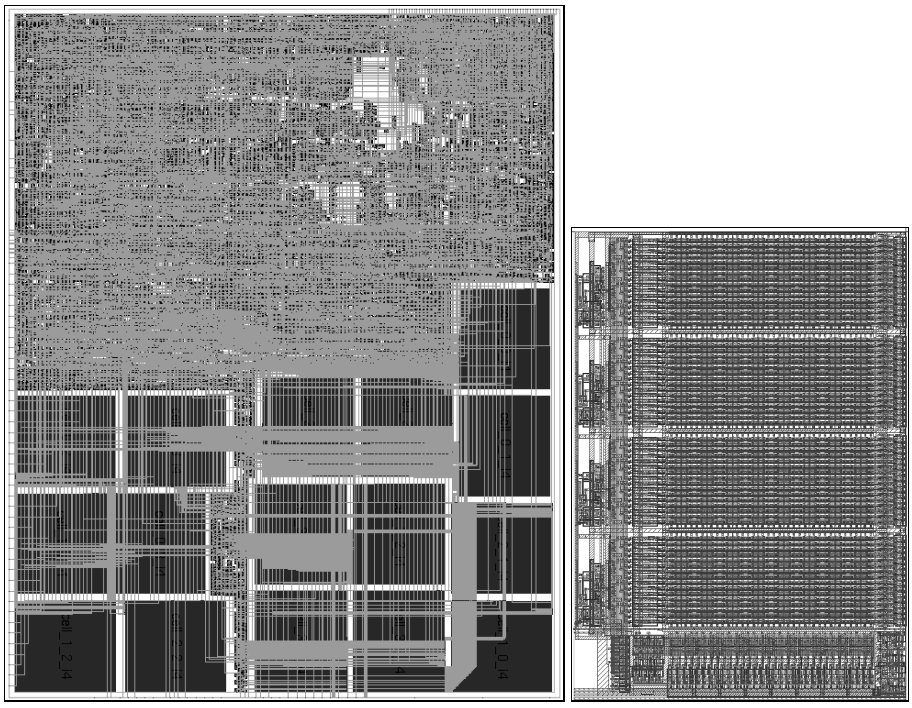


Four processors macroblock representing 16 pixel data *8-bit SAR ADC* *Sensory block*

Figure 18. The three units of the separated sensor-processor array architecture. The images are not to scale, the real sizes are 120*120, 18*80, 9*9 square microns, respectively.

The third example of processor implementation has been prepared by not only full custom design, but also using high capacity synthesis tools. The whole array has been clustered for 8*8 processor blocks. Each of the blocks contains control signal pipeline stages and intra-block buffering. The performance tradeoff of shared units is again applied in the same form as in the other examples: four pixels are represented by a processor and four memory blocks use the same decoder logic (see Figure 19.). The resulting operation speed for a complete array is the 180 MHz (post layout, backannoted simulation, worst case conditions). In this example the most area consuming portion, the memories, are remained full-custom designs using 3T DRAM cells. The other elements, ADC and sensors, was the same as in the second example.

The concluding remarks on the experiments is that first, the former estimates holds for these cases, and second, the various design techniques and methods has been confirmed the feasibility of the architectures.



*8*8 processor array* *1Kbit 3TDRAM memory block*

Figure 19. A block of synthesized processor array of 64 processors representing 256 pixels (on the left side). The black boxes are full-custom memory macros (on the right side). The technology is 0.18 micron, VST standard cell library, the size of the block and the memory are 880*700, 85*120 micron square, respectively.

VII. CONCLUSIONS

In this paper, we presented the digital implementation method of the cellular sensor-computer. As can be seen, the continuous, run-time, reconfiguration of the small number of bit-serial resources yields a highly compact, but still versatile and high performance architecture. We highlighted some interesting and distinguishing properties of the integration of sensing and processing. We showed also that the physical separation of sensing and processing arrays with proper interconnection yields almost equivalent performance than they tight integration.

ACKNOWLEDGMENT

This research was founded by the Grant of the National Science Fund of Hungary (OTKA), the multidisciplinary doctoral school at the Faculty of Information Technology of the Pázmány P. Catholic University and the Office of Naval Research (ONR) Grant No. N0001-4021-0884 and the European Community (Grant No. IST-2001-38097 LOCUST).

REFERENCES

[1] T.Roska and L.O.Chua, "The CNN Universal Machine: An Analogic Array Computer", *IEEE Trans. Circuits and Systems,* Ser.II., vol. 40, pp. 163-173, 1993

[2] T. Roska, "Computer-Sensors: Spatial-Temporal Computers for Analog Array Signals, Dynamically Integrated with Sensors", *Journal of VLSI Signal Processing,* Vol.23, pp.221-237, 1999

[3] T. Roska and Á. Zarándy: "Proactive Adaptive Cellular Sensory-Computer Architecture via extending the CNN Universal Machine", to be published on the *ECCTD 03 European Conference on Circuit Theory and Design*, 1 - 4 September 2003, Kraków, Poland

[4] A. El Gamal: “High Dynamic Range Image Sensors”, Tutorial at *International Solid-State Circuits Conference*, February 2002, Available: *http://www-isl.stanford.edu/~abbas/group/papers_and_pub/isscc02_tutorial.pdf*

[5] X. Q. Liu and A. El Gamal, “Photocurrent estimation from multiple nondestructive samples in a CMOS image sensor,” *SPIE2001*, Proceedings, pp: 4306-4310, Jan. 2001.

[6] G. Linán, P. Földessy, S. Espejo, R. Domínguez-Castro and A. Rodríguez-Vázquez, “A 0.5µm CMOS 10^6 Transistors Analog Programmable Array Processor for Real-Time Image Processing”. *Proc. of the 25th European Solid-State Circuits Conference*, pp. 358-36, Duisburg-Germany, Sept. 1999.

[7] P.Dudek and P.J.Hicks, "A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip", *IEEE Transactions on Circuits and Systems - I: Analog and Digital Signal Processing*, vol. 520, no. 1, pp. 13-20, January 2005

[8] G. Linan: “ACE16K: an Advanced Focal-Plane Analog Programmable Array Processor”, *ESSCIRC 2001 Presentations 27th European Solid-State Circuits Conference*, Villach, Austria, 18-20 September 2001

[9] “Bi-i Real-time Intelligent Cameras”, Analogic-computers Ltd. whitepaper, available at http://www.analogic-computers.com/ProdServ/Bi-i

[10] T. Hamamoto and K. Aizawa: “A Computational Image Sensor with Adaptive Pixel-Based Integration Time”, *IEEE Journal of Solid State Circuits*, Vol. 36. no. 4 April. 2001

[11] M. D. Grossberg and S. K. Nayar: “High Dynamic Range from Multiple Images: Which Exposures to Combine?”, *Int. Proc. ICCV Workshop on Color and Photometric Methods in Computer Vision (CPMCV)*, Nice, France, October 2003.

[12] S. K. Nayar and T. Mitsunaga: “High Dynamic Range Imaging: Spatially Varying Pixel Exposures”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, June 2000.

[13] Bolotski, M. Abacus: A Reconfigurable Bit-Parallel Architecture for Early Vision. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[14] Murray, A.F.; Denyer, P.B., „A CMOS Design Strategy for Bit-Serial Signal Processing”, *IEEE Journal of Solid-State Circuits*, Vol 20, Issue 3, pp: 746 – 753., June 1985.

[15] J. Serra, Image Analysis and Mathematical Morphology. New York: Academic, 1982.

[16] Diamantaras, K.I.; Zimmermann, K.H.; Kung, S.Y., ”Integrated fast implementation of mathematical morphology operations in image processing”, *IEEE International Symposium on Circuits and Systems*, Vol.2, pp. 1442 – 1445, 1-3 May 1990

[17] M.C. Herbordt, J.B. Cravy, R. Sam, O. Kidwai, C. Lin, “A System for Evaluating Performance and Cost of Massively Parallel Array Designs”, *Journal of Parallel and Distributed Computing*, No. 60 (2), pp. 217-246.

[18] L. Wanhammar, DSP Integrated Circuits, Academic Press, 1998.

[19] R. Wagner, Á. Zarándy and T. Roska “Adaptive Perception with Locally-Adaptable Sensor Array”, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 51., No. 5., pp: 1014 - 1023, May 2004.

[20] A. Zarándy, P. Földesy, T. Roska, “Per-pixel integration time controlled image sensor”, *European Conference on Circuit Theory and Design*, *ECCTD2005*, Cork, Ireland, August 31-Sept 1, 2005. accepted for publication.

[21] Ohta, A.; Isshiki, T.; Kunieda, H., „New FPGA architecture for bit-serial pipeline datapath”, *IEEE Symposium on FPGAs for Custom Computing Machines*, Proceedings, pp: 58 – 67, 15-17 April 1998.

[22] Rudack, M.; Niggemeyer, D., „Yield enhancement considerations for a single-chip multiprocessor system with embedded DRAM”, *International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT '99*, pp 31 – 39, 1-3 Nov. 1999.

[23] Georges G. E. Gielen, Wim Dehaene, Phillip Christie, Dieter Draxelmayr, Edmond Janssens, Karen Maex, Ted Vucurevich, “Analog and Digital Circuit Design in 65 nm CMOS: End of the Road?”, *Design, Automation and Test in Europe* (DATE2005), pp. 36-42, March 7-11, Munich, Germany.

[24] Hui Tian; Fowler, B.; Gamal, A.E.,”Analysis of temporal noise in CMOS photodiode active pixel sensor”, *IEEE Journal of Solid-State Circuits*, Volume 36, Issue 1, pp. 92 – 101, Jan. 2001.

[25] Peter B. Catrysse, Brian A. Wandell, „Optical efficiency of image sensor pixels”, *JOSA*, Volume 19, Issue 8, 1610-1620, August 2002.

[26] El Gamal, A., “Trends in CMOS image sensor technology and design”, *Electron Devices Meeting, IEDM '02*, Digest. International, pp. 805- 808, 2002.

[27] B. Schneider et al., “Image sensors in TFA (thin film on ASIC) technology,” in Handbook on Computer Vision and Applications. Boston, MA: Academic, 1999.

[28] Stephan Benthien et al., “Vertically Integrated Sensors for Advanced Imaging Applications”, *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 7, pp. 939-946, July 2000.

[29] “Pixim Digital Pixel System”, Pixim Inc., 2005.

[30] Foldesy P., “Trends in design of massively parallel coprocessors implemented in digital ASICs”, *IEEE International Joint Conference on Neural Networks*, Proceedings, Vol. 4, pp.: 3131 – 3135., 25-29 July 2004.

[31] Robert Johansson, Leif Lindgren, Johan Melander, and Bjrn Mller, “A Multi-Resolution 100 GOPS 4 Gpixels/s Programmable CMOS Image Sensor for Machine Vision”, *IEEE Workshop on Charge Coupled Devices And Advanced Image Sensors*, 2003.

[32] Tonia Morris, Erica Fletcher, Cyrus Afghahi, Sami Issa, Kevin Connolly, Jean-Charles Korta. "A Column-based Processing Array for High-speed Digital Image Processing", IEEE *ARVLSI*, vol. 00, no. , p. 42, 20th 1999.

[33] S. Kleinfelder, SukHwan Lim, X. Liu, and A. E. Gamal,, “A 10 000 Frames/s CMOS Digital Pixel Sensor”, *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 12, pp. 2049- 2059., December 2001.

[34] D. Sinno, “Attentive Management of Configurable Sensor Systems”, Ph.D. thesis, Arizona State University, May 2000.

[35] L. Li, D. Cochran, and R. Martin, ”Target Tracking with an Attentive Foveal Sensor”, *Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, pp. 182-185, 2000.

[36] Rekeczky, C.; Szatmari, I.; Balya, D.; Timar, G.; Zarandy, A., “Cellular multiadaptive analogic architecture: a computational framework for UAV applications”, *IEEE Transactions on Circuits and Systems I*, Volume 51, Issue 5, pp. 864 – 884., May 2004.

[37] www.ti.com