A SIMD Cellular Processor Array Vision Chip With Asynchronous Processing Capabilities

Alexey Lopich, *Member, IEEE*, and Piotr Dudek , *Senior Member, IEEE*

***Abstract*—This paper describes an architecture and implementation of a digital vision chip that features mixed asynchronous/synchronous processing techniques. The vision chip is based on a massively parallel cellular array of processing elements, which incorporate a photo-sensor with an ADC and digital processing circuit, consisting of 64 bits of local memory, ALU, flag register and communication units. The architecture has two modes of operation: synchronous SIMD mode for low-level image processing based on local pixel data, and continuous-time mode for global operations. Additionally, the periphery circuits enable asynchronous address extraction, fixed pattern addressing and flexible, random access data I/O. A 19 $\times$ 22 proof-of-concept array has been manufactured in 0.35 μm CMOS technology. The chip delivers 15.6 GOPS for binary and 1 GOPS for grayscale operations dissipating 26.4 mW, while operating at 2.5 V and 75 MHz clock. Experimental measurements indicate that the presented concept favorably compares with other digital and analog vision chips. The results of low- and medium-level image processing on the chip are presented.**

***Index Terms*—Asynchronous image processing, cellular processor array, smart sensor, vision chip.**

I. Introduction

SINCE THE early developments of massively parallel processor arrays [1], [2], these systems were considered ideally suited to image preprocessing applications, which are characterized by regular, local computations with inherent pixel-level parallelism. More recently, massively parallel processing has been also used in devices integrating image sensing and processing into a single silicon device, which has led to the development of so-called vision chips [3], [4]. The architecture of these devices is based on fine-grain processor-per-pixel arrays, where each cell combines a photosensor and a processor (Fig. 1). Such an approach benefits from eliminating the I/O bottleneck as the sensory data is fed directly into the corresponding processing elements. Vision chips also benefit from small silicon area and reduced power consumption as compared to conventional vision systems with separate sensing and processing units.

So far, the field of application for such devices was primarily low-level image processing, where the input data is represented by a 2-D array of pixel values and so is the output result. Every pixel of the result image is explicitly expressed as a function of its own value and its bounded neighborhood. Consequently, the main feature of low-level image processing is pixel-level parallelism, where identical operations are performed on every pixel. Such algorithms can be naturally mapped onto processor-per-pixel arrays and the performance can improve by a factor of N^2 for $N \times N$ arrays as compared to serial architectures. From this perspective, the single instruction multiple data (SIMD) paradigm, where all the processors execute the same instructions while processing different data, provides a suitable solution to implementing a general-purpose programmable vision device.

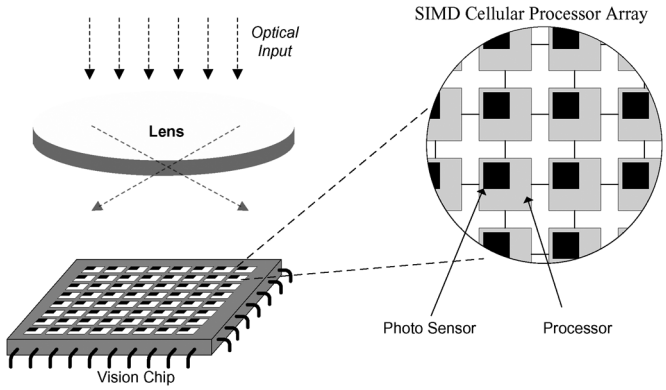

Fig. 1. A vision chip placed in the focal plane of the optical system uses a pixel-parallel array to sense and process incoming images.

However, in image preprocessing there are a number of operations that are based on an iterative process, which results in a global data flow across the pixel grid (e.g., flood-filling, skeletonization, distance transformation, segmentation, etc.). In such operations the result in each cell, despite processing only local data, implicitly depends on the entire array of pixels. Although such algorithms are feasible on a SIMD architecture, the iterative data-dependant process would result in inefficient power and time consumption when executed in a synchronous manner on such hardware. Our attention was, therefore, drawn to a challenge of how to enable efficient execution of global operations on a simple and compact parallel architecture. The successful solution would help to exploit the fine-grain parallelism more fully and significantly simplify the development of medium-level processing algorithms, thus extending application of smart sensors from preprocessing to more complex image analysis.

A number of general-purpose vision chips that employ the SIMD paradigm have been presented [5]–[11]. While it is clear that application-specific arrays [12]–[15] can provide better performance to area/power ratio, here we discuss only programmable general-purpose chips that can fulfill a wide

Manuscript received May 21, 2010; revised September 05, 2010, November 07, 2010, and January 28, 2011; accepted February 13, 2011. Date of publication April 21, 2011; date of current version September 28, 2011. This paper was recommended by Associate Editor M. Delgado-Restituto.

The authors are with the School of Electrical and Electronic Engineering, University of Manchester, Manchester, M60 1QD, U.K. (e-mail: a.lopich@manchester.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2011.2131370

range of image processing tasks. All vision chips can be categorized according to the domain in which they process image data: analog and digital. The major advantage of analog architectures is the higher utilization of area and lower power consumption with respect to achieved performance, which has been demonstrated in devices such as the SCAMP [5], ACE16k [8], MIPA4k [10], etc. In contrast to analog vision chips, their digital counterparts operate with image data represented in digital format, which provides robustness and an absolute accuracy, limited only by the word length. The major challenge of building digital processor-per-pixel arrays is the increased design size, as the number of transistors required to implement arithmetic operations is generally larger than that in analog circuits. As strict area constraints limit the amount of logic inside the PE and correspondingly its functionality, the majority of reported digital vision chips operate with 1-bit data. For example, the NSIP chip [6] contains a simple ADC, logical unit and 8-bit memory in every pixel cell. Although cells operate with binary data, simple grayscale operations are possible thanks to A/D conversions in the temporal domain. A similar approach is implemented in the PVLSAR chip [9], where a binary focal-plane processor array allows some gray-level morphological operations during the image acquisition. The latest version of the PVLSAR device [16] incorporates a 200 $\times$ 200 array of processing cells with increased internal memory of 48 bits. Another 16 $\times$ 16 array binary vision chip (SRVC) described in [11] has 30×40 μm^2 pixel pitch, where each cell can store 4 bits of data and perform basic logic operations. A 64 $\times$ 64 digital vision chip presented in [7] contains PEs with a 3 $\times$ 8-bits memory, an adder, latches and a photo-sensor with an ADC, and a cell pitch of 67.4 μm.

Although current analog vision chips [5], [8] are more area- and power-efficient then the digital ones, the progress in CMOS fabrication process and poor scalability of analog circuits reduce their advantage making digital design more favorable for future massively parallel architectures. This paper presents the architecture design and implementation of a digital vision chip, based on asynchronous/synchronous processor array (ASPA). A proof-of-concept chip, consisting of a 19 $\times$ 22 array has been fabricated. The ASPA chip is a general-purpose device that is based on the architecture outlined in [17] and partially described in [18]. In this work we present full details of the hardware implementation, performance evaluation and experimental results. Each processing cell of the ASPA chip is based on universal synchronous digital architecture, but also enables generation and processing of asynchronous locally controlled trigger-waves [19], continuous-time execution of a distance transformation and global data transfers. A mixed asynchronous/synchronous approach facilitates efficient execution of both local and global operations in a compact massively parallel architecture. Experimental results indicate that the chip delivers real-time performance not only in low-level image preprocessing, but also in more complex medium-level image analysis tasks.

II. Architecture

The architecture of the proposed chip is presented in Fig. 2. It consists of a 2-D array of locally interconnected asynchronous-synchronous processing elements (PEs), placed in a

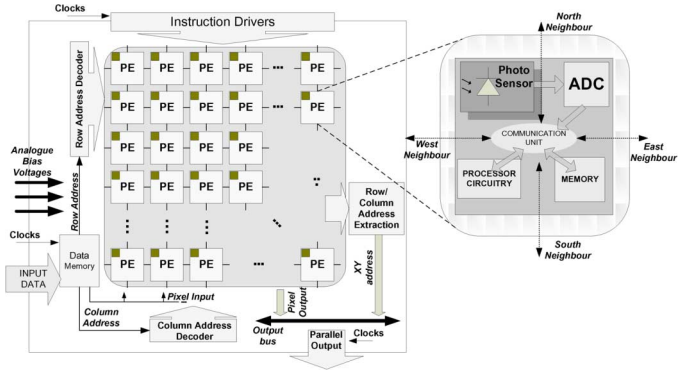

Fig. 2. ASPA chip architecture.

4-connected rectangular grid. Every PE is connected to its four neighbors in such a way that data exchange in both discrete- and continuous-time domains is supported. The program for all processors is stored remotely in the memory of a central controller. The instruction word (IW) is broadcast by the controller to every PE cell, so that all PEs receive identical instructions and execute them in parallel (SIMD mode), while some degree of local autonomy is provided by activity "flag" indicators. The IWs and local flags are also used to configure the array for continuous-time operation.

Apart from the processor array the device comprises instruction drivers, control logic and addressing circuitry for random-access input/output. The purpose of instruction drivers is to distribute control signals across the array with a minimum skew. Addressing circuitry enables random pixel access, as well as block and fixed pixel pattern addressing [20]. Additional functional units are introduced for pixel address extraction, enabling address event representation (AER) readout mode [21]. To output various data to the host controller, dedicated circuitry multiplexes this data into output buffers.

A. Processing Element

The architecture of the cell is determined by functional requirements, which in turn are imposed by processing algorithms that are expected to be implemented on the array. The preprocessing algorithms can be divided into two groups. The first group involves operations based on a priori determined neighborhood (e.g., convolutions, morphological operators, etc.). These algorithms are based on either arithmetic or logic operations. Hence, these operations have to be supported within each cell. The second group is characterized by the dependence of the result in each pixel on the entire image. These operations can often be represented as an iterative process, where a pixel value, determined as a function of local neighborhood on each iteration, implicitly depends on the entire array. In each iteration the actual useful processing is performed in a limited number of pixels, placed at the front of a wave-propagating activity and therefore synchronous iterative realization of these operations on a SIMD array would result in inefficient power consumption and reduced performance. Since the execution of such operations depends entirely on the data flow, asynchronous methodologies can be employed. However every asynchronous algorithm has a unique hardware realization and therefore it is

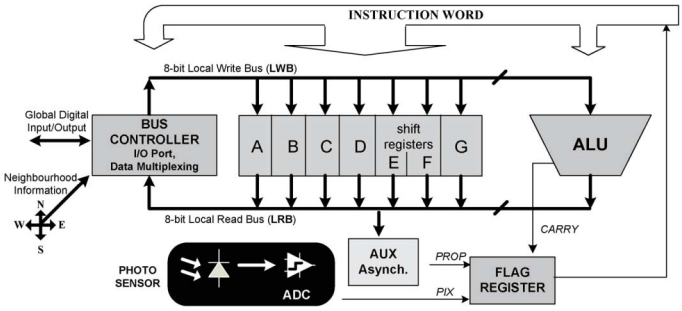

Fig. 3. The block diagram of the processing cell.

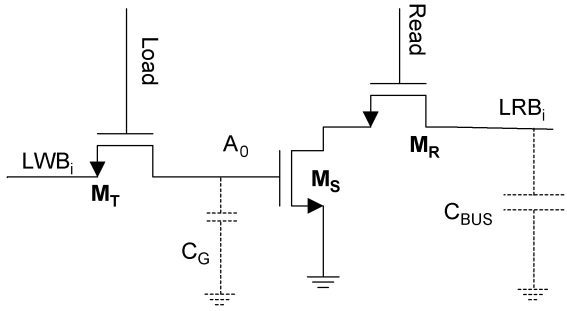

Fig. 4. Basic memory cell—dynamic latch.

necessary to identify those fundamental operations that can be used as a basis for the majority of global algorithms.

In order to facilitate the execution of both local and global operations, the organization of a basic cell is similar to that of a simple microprocessor with added functionality to support global asynchronous data-processing. The block-diagram of the processing cell is presented in Fig. 3. The architecture comprises an I/O port, digital arithmetic and logic unit (ALU), bus controller (BC), register file, flag register, photo-sensing unit, and auxiliary circuitry. The processor operates with digital values in a discrete-time mode, but can also be configured to operate in a continuous-time mode.

The operational principle of the cell is based on register-transfer operations. A received instruction specifies what type of register-transfer operation is to be performed: for arithmetic operations—transfer between the memory and the ALU or a shift register; for I/O operations–transfer between memory and global I/O port; for neighborhood communication–transfer between memory registers in neighboring cells (*North*, *East*, *South*, *West*). In addition, the instruction set is completed with conditional branching and global asynchronous operations.

Since the majority of low-level image processing algorithms are based on operating on local neighborhood data, local data transfers contribute a significant part to the computational load. The speed of local neighborhood data transfer operations is therefore an important factor. The most optimal way to extract local neighbors' data is to have an immediate access to their internal memory. Thus, by applying appropriate control signals to select the required information source (e.g., local data, neighbor's data, or globally broadcast data) the procedure of accessing data in local neighbors is identical in terms of clock cycles to a register-transfer operation within a pixel.

Seven 8-bit digital general-purpose registers (GPRs), including two shift registers, have been implemented in the presented design (A, B, C, D, E, F, G). In addition, an accumulator register from the ALU can also be used as an 8-bit memory storage cell. Thus, every PE is capable of storing eight bytes of information, which is a sufficient amount of memory to execute many low- and medium-level image processing algorithms. It is also possible to store and process several values shorter than 8-bit in the same register (e.g., two 4-bit values or eight 1-bit values). This feature provides flexibility of the 64-bit memory usage for bit-serial and bit-parallel processing.

To enable data manipulation during transfers, and to increase the throughput of pixel communication, all data-transfer operations are performed in a bit-parallel manner. However, all arithmetic calculations are performed in a bit-serial manner. On one hand, such an approach decreases operational performance as compared to bit-parallel operation. On the other hand the dependence of the ALU size (and carry propagation effect) on the width of the processed data is eliminated and logic size is minimized.

III. Circuit Implementation

A. Local Memory and Datapath

In order to minimize the physical PE size the internal memory circuits are based on dynamic logic. A simple one bit latch is based on a 3 transistor dynamic memory cell (Fig. 4). When the switch $\mathrm{M_T}$ is closed, the storage capacitance $\mathrm{C_G}$ is either charged or discharged depending on input data bit i from a Local Write Bus (LWB_i) The output of the cell is connected to the corresponding bit of the precharged local read bus (LRB_i). Logic levels on the LRB are sensed by inverters that drive inputs of the bus controller. If both values *Read* and A_0 are logic "1" then the transmission path to ground is formed by closed switches $\mathrm{M_S}$ and $\mathrm{M_R}$ so that capacitance $\mathrm{C_{BUS}}$ is discharged, otherwise the value at the output node LRB_i is preserved. Transistor sizes in the memory elements were minimized ($\mathrm{W/L} = 1\ \mu\mathrm{m}/0.35\ \mu\mathrm{m}$) in order to optimize the area. The storage node capacitance $\mathrm{C_G}$ (4 fF) is formed between the gate of the $\mathrm{M_S}$ and ground plane. The leakage of the stored charge due to the reverse-bias and subthreshold leakage currents will eventually corrupt the stored value. Although retention time can be significantly improved by decreasing W/L for $\mathrm{M_T}$ and partly by increasing $\mathrm{C_G}$, the prime goal of this work to optimize the cell area, therefore the circuitry is adjusted to minimize the footprint. The measured memory retention time is 91.3 ms for 3.3 V operating voltage and 69.9 ms for 2.5 V (at a typical luminance level), which is higher than standard video frame rate (40 ms at 25 frames/s) and, if required, simple data refreshment can be performed.

One bit-slice of the 8-bit BC is shown in Fig. 5. The BC is used to multiplex internal and external data onto the LWB as well as perform bit-wise logic OR on input data. The inputs to the BC are a global input ($GLOB_i$), inverted LRB signal (INT_i), and inverted LRB signals from the four neighbors (S_i, N_i, W_i, E_i). Control signals ($sel_{GLOB}, sel_{INT}, sel_S, sel_N, sel_W, sel_E$) specify the value to be transferred to the LWB, thus enabling the following set of operations: GPR $\rightarrow$ GPR (ALU), Global I/O Bus $\rightarrow$ GPR (ALU), Neighbor $\rightarrow$ GPR (ALU).

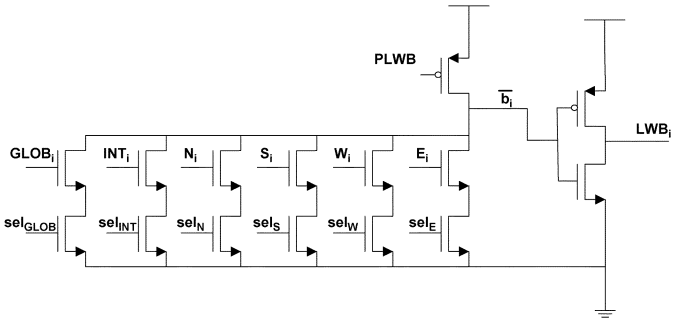

Fig. 5. One bit slice of the 8-bit bus controller (BC).

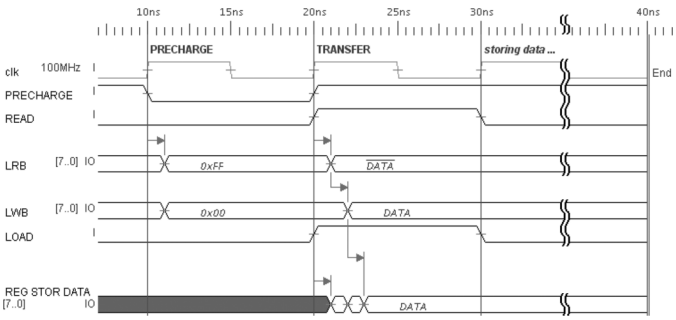

Fig. 6. Timing diagram for two-phased register-transfer operation.

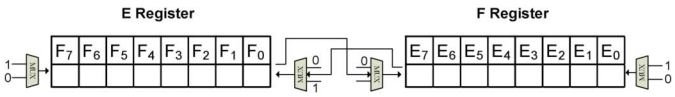

Fig. 7. Shift register interconnection.

The basic register-transfer operation comprises two clock cycles: *precharge* and *transfer* (Fig. 6). During the precharge cycle the LRB and an internal node $\overline{b_i}$ in the bus controller are being precharged so that values on the LRB and the LWB are $0xFF$ and $0x00$ correspondingly. During the transfer cycle the data is read from any register(s) or accumulator to the LRB, transferred through the BC to the LWB and then loaded into another register(s) or accumulator. At the end of the transfer cycle, when the *LOAD* signal goes to logic "0," the data is latched in the dynamic register. The communication between local neighbors is identical to local register-transfer operations and also performed within two clock cycles.

In addition to standard memory, there are two bidirectional shift registers. Based on a modified memory cell these registers allow data shift in both directions with a shift-in signal. Memory cells assigned to store most and least significant bits (MSB and LSB) can be reconfigured to allow the two registers to act as a single 16-bit register, i.e., it is possible to shift data in two registers separately, as well as simultaneously so that the MSB of register F is loaded into the LSB of register E, and vice versa (Fig. 7). This feature enables efficient execution of global asynchronous distance transforms, unsigned multiplication, and division.

II. ALU

The block-diagram of the ALU is presented in Fig. 8. It consists of an input multiplexer, full subtractor (SUB), two dynamic flip-flops for storing *carry* (ACC_C) and temporary single bit result (ACC_R) and demultiplexer combined with accumulator for the final result (ACC).

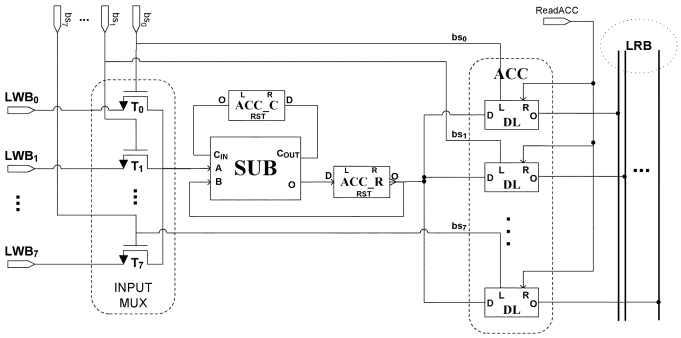

Fig. 8. ALU architecture.

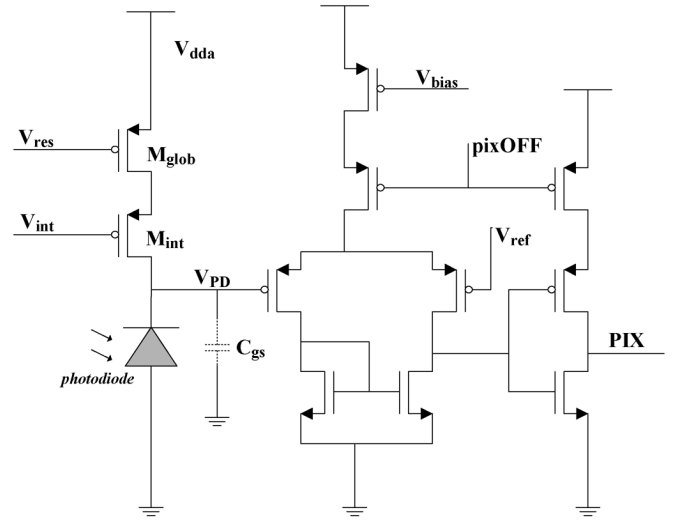

Fig. 9. Photosensor circuit.

The input multiplexer is connected to the LWB. Control signals $(\mathrm{bs}_0, \ldots, \mathrm{bs}_7)$ specify which bit is currently being processed.

The operation of the ALU is based on binary subtraction. In order to perform binary addition, a subtraction of one operand from zero is required to form a negative number.

C. Photo Sensor

A photo sensing circuit in every PE is based on a photodiode followed by a simple voltage comparator (Fig. 9). The photodetector works in an integration mode. The voltage across photodiode $\mathrm{V_{PD}}$ is compared to the reference voltage $\mathrm{V_{ref}}$. The inverted value of the comparator output is transferred to the *PIX* input of the flag register. Based on this flag indicator a digital value that corresponds to light intensity can be written to a corresponding GPR. For gray-level conversion the binary threshold is performed at different times and/or with multiple $\mathrm{V_{ref}}$ levels [22]. Current A/D conversion is set, but not limited, to 8-bit resolution. Ultimately, dynamic range will be limited by transistor matching in the comparator as well as integration time. Signal *pixOFF* is used to turn-off the biasing currents when the sensor is not accessed. Due to the usage of in-pixel resources (e.g., flag register and GPR) during sensing and digital conversion, the actual data processing is performed after the frame is sensed. Although it is not optimal compared to parallel sensing and processing (previous frame), such configuration mitigates the effect of digital processing on sensor performance.

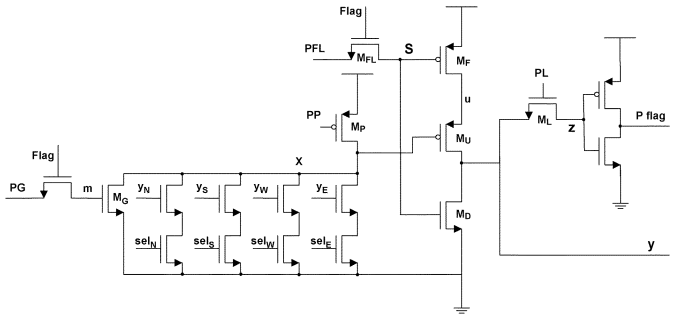

Fig. 10. Asynchronous propagation cell.

During the reset phase $\mathrm{V_{res}}$ and $\mathrm{V_{int}}$ are set to logic "0" so that switches $\mathrm{M_{glob}}$ and $\mathrm{M_{int}}$ are closed and capacitance $\mathrm{C_{PD}}$ is charged with $\mathrm{V_{PD}} = \mathrm{V_{dda}}$. While $\mathrm{V_{res}}$ is broadcast globally to all PE's at the same time, $\mathrm{V_{int}}$ is a locally controlled signal connected to LRB_7. Such configuration enables local control over pixel integration time.

D. Asynchronous Propagation Cell

The asynchronous propagation cell is designed to provide a simple tool for binary trigger-wave propagations across the pixel array in order to perform global morphological operations such as object reconstruction and hole filling in a fast and power-efficient way [19]. Fig. 11 illustrates initial steps of propagation and demonstrates an example of closed curve detection algorithm. When running binary trigger-wave propagations, all pixels in image I are divided into three sets: S_P—propagation space (pixels that can be involved in the propagation); S'_P—non-propagation space (a set complement to $S_P, S'_P = I \setminus S_P$); S_M—set of initial markers (pixels that initiate propagation). The propagation starts from the initial corner marker S_M and spreads across the propagation space S_P (white pixels). Black pixels (representing objects' borders S'_P) prohibit the propagation, thus leaving the internal space of closed contours "untriggered." This propagation splits the entire array into two subsets [Fig. 11(e)], which consist of pixels that have been involved in the propagating activity (light gray) and pixels that remain in the original state (white).

The schematic of the asynchronous cell is illustrated in Fig. 10. The initial binary image [Fig. 11(a)] is supplied to the input PFL and then conditionally drives node u. Initial state [Fig. 11(b)] is loaded to m. The propagation cell outputs its current state y. Values y_N, y_E, y_S and y_W represent the current state of the nearest neighbors, whereas $\mathrm{sel_N}$, $\mathrm{sel_S}$, $\mathrm{sel_W}$ and $\mathrm{sel_E}$ are masking signals. The output function is as follows:

$$y = (y_N \cdot sel_N + y_E \cdot sel_E + y_S \cdot sel_S + y_W \cdot sel_W + PG) \cdot PFL \cdot \mathrm{Flag}. \quad (1)$$

The propagation across the array resembles a domino effect and is carried out in an asynchronous manner with a subnanosecond delay per PE. Masking signals correspond to four least significant values of the LRB, i.e., $sel_N = \mathrm{LRB_3}$, $sel_W = \mathrm{LRB_2}$, $sel_S = \mathrm{LRB_1}$, $sel_E = \mathrm{LRB_0}$. Thus, the direction of propagation as well as pixel network topology is

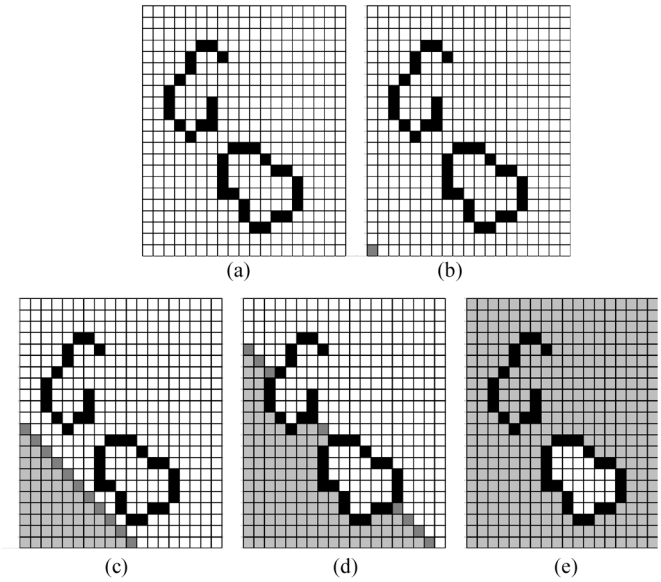

Fig. 11. Basic steps of executing asynchronous propagation on ASPA: (a) Propagation space S_P is defined (white pixels). (b) Set of initial markers $S_M (S_M \subset S_P)$ is defined (dark gray pixels). (c), (d) Propagation starts from initial markers S_M and runs through entire array with useful processing occurring only in the wave front pixels (marked by dark gray), whereas the rest of the array is idle.

flexibly controlled from within each individual PE. Inputs PG, PFL and PL are controlled by the flag register.

The circuit is based on the assumption that $S_M \subset S_P$. The operation of the circuit consists of initialization and propagation phases. During the initialization phase, the node x is precharged through the $\mathrm{M_P}$ switch by setting PP into logic "0" and node y is discharged through $\mathrm{M_D}$ to ground by applying logic "1" to PFL. At this stage m, sel_N, sel_S, sel_W, and sel_E are set to logic "0." The second step is setting the propagation space. It is done by setting S to logic "0" so that u switches to logic "1." The $Flag$ value is now set to "1" only for propagation space [Fig. 11(a)]. Subsequently PFL is switched off from "1" to "0." For nonpropagation space, switch $\mathrm{M_{FL}}$ is open and node S remains at logic "1." The propagation is initiated by applying logic "1" to PG [Fig. 11(b)]. The only possible transition at this state for a $Flag$ value is "1" → "0," otherwise the propagation space that has been set previously would be distorted. Taking into consideration the fact that PG is also controlled by the $Flag$ value the set of propagation markers is expected to be a subset of $\mathrm{S_P}$. Applying the marker leads to node x being discharged through $\mathrm{M_G}$, which in turns results in closing the switch $\mathrm{M_U}$. The output y is then switched to "1" only if the switch $\mathrm{M_F}$ is closed, i.e., the pixel belongs to a propagation space. Any cell is triggered by another neighbor cell through one of the inputs y_N, y_E, y_S or y_W. The further process is controlled by corresponding signals sel_N, sel_S, sel_W or sel_E. If $sel_x \cdot y_N =$ '1' then the node x is discharged and propagation continues in domino-effect fashion according to the scenario described above. After propagation is finished the result is stored at node Z of a dynamic latch. The inverted value can be then read and processed in the flag register.

The power consumption of the circuit is minimized since there are no static currents. The energy required for circuit operations is used to charge and discharge parasitic capacitance

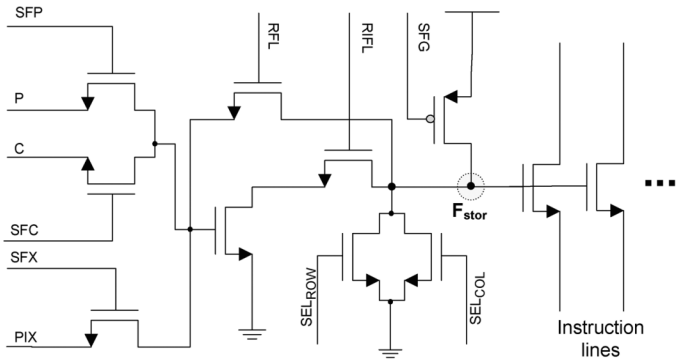

Fig. 12. Flag register.

at nodes x, y and all gates of switches in the control lines. Although the circuit ensures that there is no current through the $\mathrm{M_F}, \mathrm{M_U}, \mathrm{M_D}$ chain, it is important to avoid current through the $\mathrm{M_P}, \mathrm{M_G}$ (or switches that correspond to neighbors) chain. This is achieved by making PP = '0' and PG = '1' a prohibited combination.

E. Flag Register

The conditional execution of broadcast instructions is implemented by introducing an activity flag mechanism. Conditional operation is based on the fact that some instruction lines contain pass transistors, which are controlled by a flag value. If the PE has flag set to logic "0" the following instruction signals are "ignored" until flag is reset back to logic "1."

The schematic diagram of the flag register is shown in Fig. 12. The condition selection is performed by selecting an appropriate flag value pulling select signals SFP, SFC or SFX high, so that the corresponding value is transferred to a storage node F_{stor}. The P input is connected to the propagation indicator, C corresponds to a carry signal in the ALU and PIX represents the output of the voltage comparator in the photosensor. The latter is used to perform A/D conversion. The flag register enables branching on both actual and complement conditions (e.g., `if(C)` and `if(!C)`). During unconditional operation the control signals SFG, RFL and $RIFL$ are set to logic "0." In this case all switches on instruction lines are closed and the PE executes all transmitted instructions. If a conditional instruction such as `if(C)` or `if(!C)` is met, then SFG,SFC, RFL or $RIFL$ are set to "1," so that value C or $!C$ appears at node F_{stor}. When executing conditional instructions, the instruction word has to be set to its initial state before reading the flag value. The pixel addressing mechanism for array readout is also based on flag register operation. If a particular PE is to be selected, then row- and column-select signals for this PE are set to logic "0," while for the rest of the cells at least one of them is to be set to "1," i.e., $\mathrm{SEL_{COL}} \vee \mathrm{SEL_{ROW}}$ = '1'. The latter results in discharging node F, meaning that these cells are "unselected" and the following instructions are not to be executed.

The design size is optimized by applying conditional execution only to register load operations $(LA, LB, LC, LD, LSE, LE, RSE, LFE, LF, RFE, LG)$.
In addition to that, the flag register controls three propagation-control instructions: setting initial propagation markers and defining the propagation space (PFL), initiating propagation (PG) and loading the propagation latch (PL). It also controls one register-read operation (to enable division operations, flexible data transfers and data output). Hence, those PEs that do not satisfy the indicated condition still perform instructions related to memory reading and arithmetic calculations, but all the "store" instructions are ignored.

Nested conditions can be realized only with complement flag values, i.e., `if(!C)` or `if(!P)`, because every such condition is expected to define a subspace of the space defined by the previous conditional operation. This is only possible when the node F is discharged since this operation can only reduce the number of previously flagged pixels.

F. Data I/O

Random access data I/O operations are supported. One dedicated output register is used to output data to the Global Data Bus (GDB) (any register can be used for input). This register is different from GPRs in the way that read operation is also controlled by the flag, i.e., only addressed pixels output data.

In practice, data from any GPR or accumulator can be output, but because they are output to the same precharged bus, the result value would be logic OR of all the outputs. This feature can be exploited to perform global (array wide) logic operations. The procedure of outputting data to the global bus is similar to local register-transfer operation, but instead of writing to a local register, inverted data from the LRB is transferred to the GDB.

In a similar way (through a precharged global bus) the binary value of the local flag can be read out to the global flag bus and then to the central controller. To increase system throughput eight binary values are output in parallel.

G. Periphery

In order to facilitate fast implementation of frequently used pixel search algorithms (which have $\mathrm{O}(\log_2 \mathrm{N})$ complexity when executed with binary tree search approach) ASPA chip has an address extraction unit at the periphery of the pixel array (Fig. 13). This circuit enables fast address extraction of pixels with smallest column and row indices previously marked by flag indicators. Complete cycle of locating full boundary box of one binary image object takes only 18 clock cycles or 240 ns at 75 MHz clock. Only one iteration is required for the actual address extraction, as the address value is extracted asynchronously.

H. Cell Layout

The chip has been fabricated in a standard 0.35 μm CMOS technology. The layout of a complete PE cell is presented in Fig. 14. The basic structures of the cell such as registers, BC, ALU, and flag register are marked.

Overall dimensions of the cell are $100 \times 117\ \mu\mathrm{m}^2$ (thus providing cell density of 85 cells/mm^2) with $11.75 \times 15.7\ \mu\mathrm{m}^2$ utilized as a photo sensor. As the main purpose of this chip was to test the processing functionality of the architecture, the sensing part was given less priority. Therefore, the achieved fill-factor of $\sim$2% is relatively small. Potentially, the area of the sensor could be extended underneath the wiring reaching 810 $\mu\mathrm{m}^2$, which would provide fill-factor of more than 7%. Further improvement

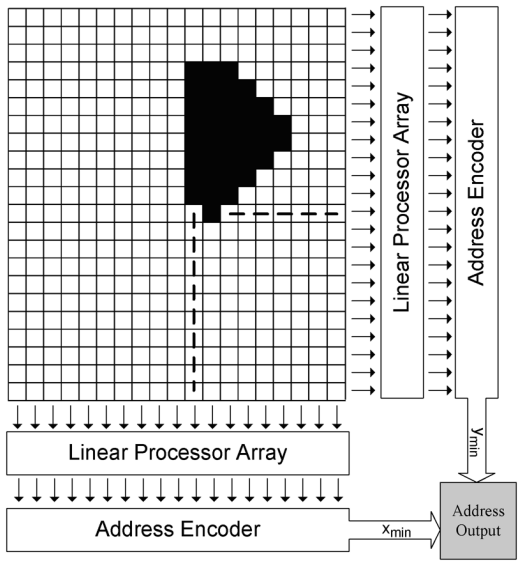

Fig. 13. Asynchronous address extraction. Periphery circuit extracts x_{min} and y_{min} coordinates of the segmented object.

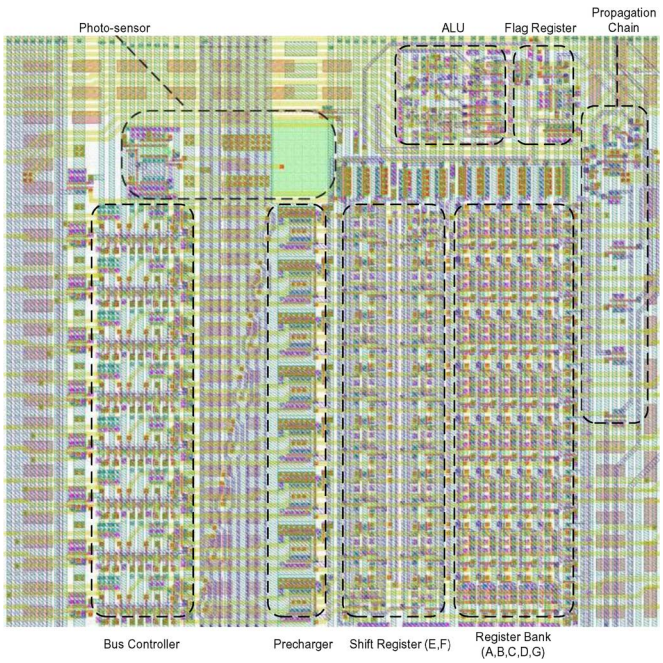

Fig. 14. Layout of the PE. The dummy structures of polysilicon and metal are inserted to adhere to fabrication rules.

of the apparent fill-factor could be achieved by placing *in situ* micro lens above each cell.

The PE circuitry consists of 460 transistors, 94% of which are n-type transistors (most of them have minimum size), which results in high transistor density, especially in memory elements. The basic memory cell occupies $3.6 \times 8.7\ \mu m^2$ so that 40 bits of internal memory (not including the shift registers and the accumulator) occupy $18 \times 70\ \mu m^2$ or only 10% of the pixel area. Since the memory elements are based on dynamic logic, the majority of charge storing nodes were shielded from the instruction lines by the ground plane to avoid cross coupling. The microphotograph of the ASPA chip is provided in Fig. 15. A 19×22 PE array has been fabricated on a 10 mm^2 prototype chip. It

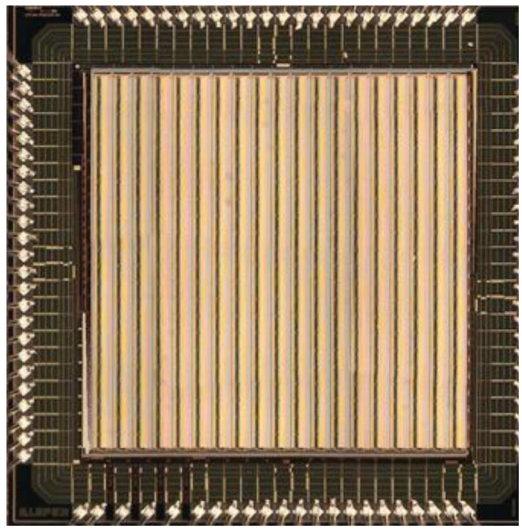

Fig. 15. Microphotograph of the ASPA.

should be emphasized that this architecture is scalable, i.e., array size may vary, although attention has to be paid to the periphery instruction drivers, since for larger arrays the load capacitance and resistance on instruction wires that go across entire array increases proportionally.

Being based on digital architecture, this design also benefits from scaling to finer technologies. Sub-100 nm technologies will require additional solutions to counter high leakage currents in dynamic registers and stronger crosstalks. However, short retention time can be addressed by a memory refresh routine, which may represent a relatively small performance sacrifice compared to the achieved benefits.

IV. Image Processing Applications

A. Local Operations

The ASPA chip has a software-programmable architecture, which makes it a truly general-purpose device, capable of executing a wide range of image processing algorithms. Every cell supports the set of basic operations (e.g., arithmetic, logic, bidirectional shift, local neighborhood data access, etc.), necessary to implement convolutions and filtering. Examples of basic preprocessing algorithms implemented in ASPA are illustrated in Fig. 16.

In order to verify the processing, images were loaded into the chip through an input interface. Subsequently the processing output results were verified versus numeric simulations. Sobel edge detection, smoothing and median filtering are performed on grayscale images in a 3×3 neighborhood. A significant number of useful morphological image processing algorithms deal with images represented in binary format (binary dilation, erosion, skeletonization). As can be noticed, ASPA can execute many of these tasks in an efficient way, as some of the logic operations are performed on data buses during data transfers, i.e., at no additional cost. For example the *closing* operation [Fig. 17(d)], where sequential dilation and erosion helps to restore object's connectivity, is executed in 130 ns when the device is clocked at 75 MHz.

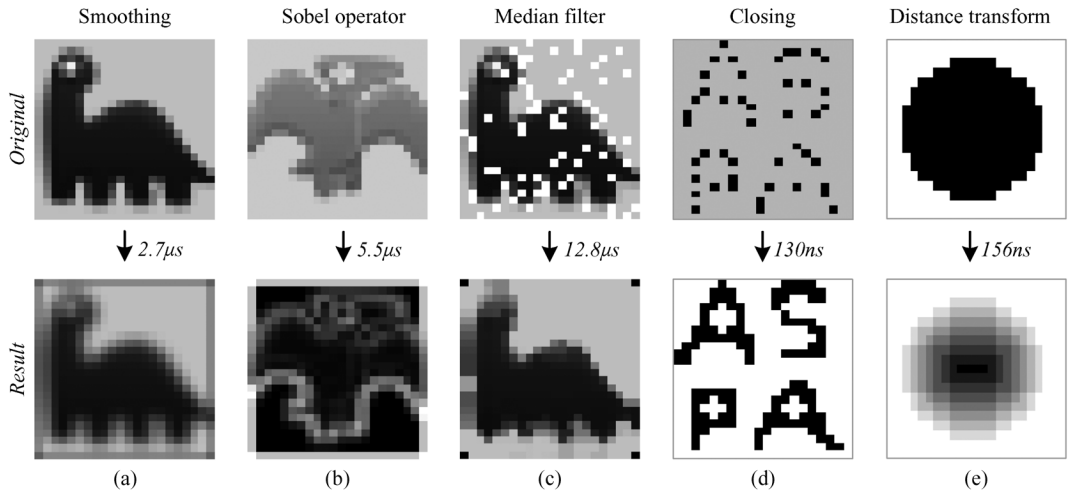

Fig. 16. Low-level image processing in ASPA. (a) Smoothing. (b) Sobel edge detection. (c) Median filter. (d) Binary closing. (e) Asynchronous distance transform.

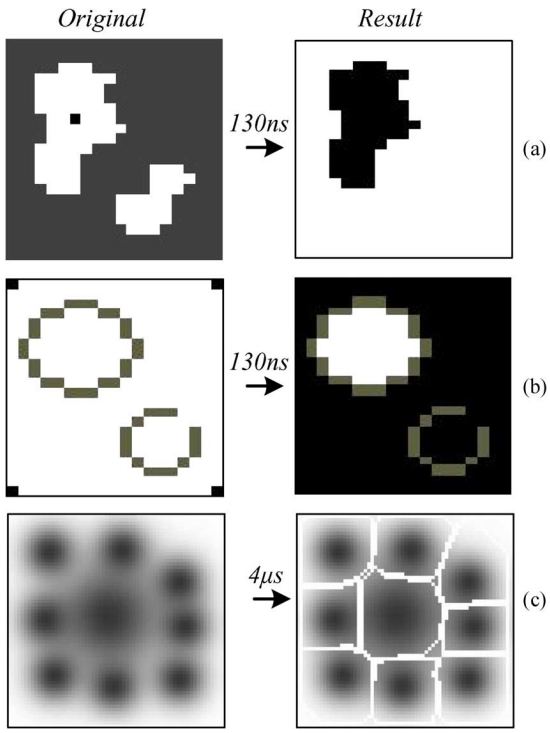

Fig. 17. Global operations on the ASPA chip. (a) Geodesic reconstruction. (b) Asynchronous hole filling. (c) Watershed segmentation (simulated).

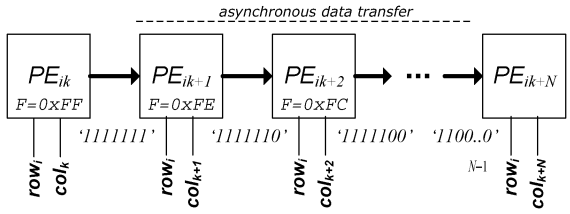

Fig. 18. Asynchronous data shift and transfer between two cells.

Register F Data		Interpreted
HEX	BIN	Distance
0xFF	11111111	0
0xFE	11111110	1
0xFC	11111100	2
0xF8	11111000	3
0xF0	11110000	4
0xE0	11100000	5
0xC0	11000000	6
0x80	10000000	7
0x00	00000000	out of range

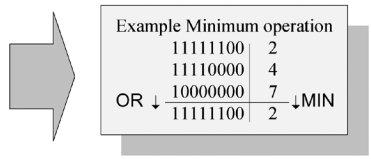

Fig. 19. Data interpretation during asynchronous transfers.

B. Asynchronous Pixel-Parallel Operation

Apart from local operations, the ASPA chip supports global binary trigger-wave propagations that can be used for geodesic reconstructions, hole filling and as part of more complex algorithms. Although such operations can easily be implemented in synchronous fashion as an iterative process, such realization is inefficient in terms of power and overall performance, as the number of required iterations should cover "worst case," e.g., $O(n \times m)$ for $n \times m$ image. With the propagation delay of 0.76 ns per pixel the equivalent synchronous implementation on ASPA in iterative manner would require operation at 2.7 GHz (two operations per cell), expecting over 1.1×10^{12} operations per second on a 19×22 array. Implementation examples of basic operations that employ binary trigger-wave propagations are presented in Fig. 17.

By utilizing the features of BC and shift registers it is possible to perform simple processing while transferring data to neighbors [23]. In particular it is possible to perform a simplified version of the minimum and increment operation. Assuming the value $0xFF$ is stored in one of two shift registers present in PE, a shift left operation will translate this value to $0xFE$ or "11111110" in binary code. By applying this operation m times $(m < 9)$ we will achieve zeroes in m least significant bits. Let us consider the chain of processing elements (Fig. 18) with the shift register F of $\mathrm{PE_{ik}}$ loaded with $0xFF$, while in other cells register F is loaded with $0xFF$. Transferring this value asynchronously through shift-registers of every cell in the chain generates distance values in cells $\mathrm{PE_{ik+1}} \ldots \mathrm{PE_{ik+N}}$, where number of zeroes in least significant bits will represent pixel distance from cell $\mathrm{PE_{ik}}$, e.g., $0 \equiv 0xFF$, $1 \equiv 0xFE$, $2 \equiv 0xFC, \ldots 8 \equiv 0x00$. Furthermore, in the context of such number representation it is easy to notice that bit-wise OR operation provides a "minimum" (Fig. 19). The bit-wise OR can be computed in the BC operating on the current pixel and its

TABLE I
BENCHMARKS FOR IMAGE PROCESSING ON DSP CHIP AND ASPA. DSP FIGURES ARE FOR TMS320DM6455 DEVICE [24] OPERATING AT 1 GHz. ASPA FIGURES ARE FROM THE FABRICATED PROTOTYPE, BUT WOULD BE THE SAME FOR A 128 $times$ 128 DEVICE

Published benchmarks [24]				Total time TMS320DM6455 scaled for 128×128 array	ASPA Ops.	ASPA Experimental processing time @75MHz, ns	ASPA Expected processing time @ 300MHz, ns
Benchmark	**Description**	**Image**	**Operation time TMS320DM6455 @1GHz, ns**				
Boundary Structural Operator	*Scans an image for non-zero pixels.*	*cols=128 rows=3*	492	**21008**	16	**416**	**96**
3x3 Binary Dilation	*Implements a 3x3 binary dilation.*	*cols=128 rows=8*	137	**2192**	5	**130**	**30**
3x3 Binary Erosion	*Implements a 3x3 binary erosion.*	*cols=128 rows=8*	137	**2192**	5	**130**	**30**
Perimeter Structural Operator	*Detects the boundary of an object in a binary image.*	*cols=128 rows=3*	135	**2160**	5	**130**	**30**
Sobel Edge Detection	*Applies horizontal and vertical Sobel edge.*	*cols=128 rows=8*	1079	**17264**	215	**5590**	**1290**
Image Threshold	*Performs a threshold operation on an input image.*	*cols=32 rows=32*	148	**2368**	18	**468**	**108**
3x3 Median Filter	*Performs 3x3 median filtering*	*cols=128 rows=3*	288	**12298**	490	**12740**	**2940**

four neighbors, i.e., the minimum is selected among the values of 4 neighbors and local data. If we assign $0xFF$ to all the background pixels, $0x00$ to foreground pixels and perform the operation described above with all 4-neighbors enabled in the BC, we will achieve a city block distance map within an 8 pixel distance range [Fig. 16(e)]. In this transformation value 0xFF corresponds to distance 0 and 0x00—to 8. If necessary the full distance map can be calculated applying the above procedure $[\mathrm{R}/8]+1$ times where R is a maximum object radius. An additional step is required to interpret the value into a binary number. If we achieve $0x00$ it indicates that processing cell B is out of range of an 8-pixel radius.

The above described basic global operations together with asynchronous address extraction can be efficiently used as a part of computationally expensive algorithms, such as skeletonization, object tracking and recognition [23]. For example a watershed transformation of 64 × 64 image with 9 catchment basins will be achieved in 4 μs at 300 MHz clock [Fig. 17(c)].

V. PERFORMANCE AND COMPARISON

The overall performance of the chip is evaluated according to the executed operation (binary, grayscale, unsigned multiplication and division) and operation mode (synchronous, asynchronous). At 75 MHz system clock every PE in the ASPA provides 37.5 MOPS (binary), 2.4 MOPS (grayscale), and 0.3 MOPS (for unsigned products and quotients) in a synchronous mode. The full 8-bit arithmetic subtraction requires 32 clock cycles, consuming 0.07×10^{-3} W per cell at 75 MHz, thus providing 0.3×10^{11} OP/J. The achieved area utilization is 205.2 MOPS/mm^2 (for grayscale operations). Postlayout simulations demonstrate that the design will operate at 300 MHz clock and performance figures will quadruple, however our test system did not allow instruction issue at such rate. Extrapolated performance characteristics for a 128 × 128 array are as follows: 0.61 TOPS for binary operations, 39.3 GOPS for grayscale operations and 4.9 GOPS for unsigned products. The simulations showed that operating at 300 MHz the chip would consume ~ 350 μW/cell. In the experimental setup at 75 MHz 19 × 22 array consumed 26.35 mW ($\mathrm{V_{DD}} = 2.5$ V), resulting in 63.04 μW/cell.

When comparing various architectures certain figures of merit (e.g., "operations per second") can be misleading because different designs can exhibit different performance characteristics depending on the target application and algorithm implementation. Therefore, performance evaluations provide more valuable information when they are based on benchmarking the target device on a fixed set of tasks. Available benchmarks for image analysis and filtering operations allowed us to compare the ASPA chip with a state-of-the-art DSP device [24]. The results are presented in Table I. The execution times are given for the implemented 19 × 22 array, however the ASPA execution times would remain the same for larger arrays (e.g., 128 × 128), due to architectural parallelism. It can be noticed, that the ASPA approach has the potential to outperform the DSP at low-level (fixed-point) image preprocessing by a significant margin despite operating at lower frequency. Although the benchmarking is provided for a 128 × 128 array, the ASPA execution times would remain the same for larger arrays as well, due to architectural parallelism. The DSP is fabricated in a 90 nm process and at 1.25 V consumes 2 W. At the same time a 128 × 128 version of the ASPA chip operating at 2.5 V would consume 1 W (extrapolated from 26.35 mW for the 19 × 22 chip).

The efficiency characteristics for various parallel processors [25]–[30] are summarized in Table II. The performance was estimated for 8-bit arithmetic operations. The first column for the

TABLE II
COMPARISON OF PARALLEL IMAGE PROCESSORS, TI DSP, AND ASPA VISION CHIP

Processor	**TMS320DM6455 [24]**	**Xetal2 [25]**	**Matrix [29]**	**Xenon v3 [26]**	**IMAP-CE [27]**	**iVisual [30]**	**Multiresolution Smart Sensor [28]**	**ASPA Experimental @75MHz**	**ASPA Expected @300MHz**
P_E, GOPS/W	**3.8**	**178**	**200**	**21.3**	**12.75**	**168**	**18.3**	**38**	**29**
P_A, MOPS/mm^2	**83.2**	**1400**	**1298**	**430**	**421.5**	**1160**	**66.4**	**205.2**	**820.8**
Process	90nm	90nm	90µm	0.18µm	0.18µm	0.18µm	0.35µm	0.35µm	0.35µm
Number of PE's	1	320	2048	64	128	128	1536	418	16384
Architecture	RISC DSP	1D SIMD Array	1D SIMD Array with 32 banks of shared memory per 64 PEs	2D 8×8 SIMD Array, 8×8 sensors per PE	1D SIMD Array	1D SIMD Array +127ALU connected as a tree + RISC processor	1D SIMD Array	2D 22×19 SIMD Array	2D 128×128 SIMD Array
Photo Sensor (PS)	n/a	n/a	n/a	n/a, 64×64 sensor integrated via bump bonding	n/a	128×128, separate from processor array	1536×512, separate from processor array	22×19, integrate with array (PS per PE)	128×128, integrate with array (PS per PE)
Target Applications	High-level	Low- and Medium -level	Low- and Medium -level	Low- and Medium-level	Low-, Medium-, High-level	Low- and Medium -level	Low-level, multi-sense imaging	Low- and Medium - level	Low- and Medium - level
Voltage, V	1.25	0.9-1.2	1.2	1.8	1.8	2.1	3.3	2.5	3.3
Frequency, MHz	1000	84	200	100	100	50	33	37.5	150

TABLE III
COMPARISON OF ANALOG AND DIGITAL VISION CHIPS

Reference	**SPE [7]**	**ACE16k [8]**	**SRVC[11]**	**SCAMP [5]**	**ASPA Experimental @75MHz**	**ASPA Expected @300MHz**
Processing type	digital	analog	digital (bin.)	analog	digital	digital
Array Size	64×64	128×128	16×16	128×128	19×22	128×128
Cell Size, µm^2	67.4×67.4	75.7×77.3	30×40	49.35×49.35	100×117	100×117
Technology, µm	0.35	0.35	0.18	0.35	0.35	0.35
Memory per cell	24 bit	8 an. reg,4 Bin	4 bits	9 an. reg.	64 bit	64 bit
Performance, GOPS	6.4	330	0.213	20	1	157
Die Size, mm^2	29.2	145.2	2.25	50	9	213.5
Power per chip	N/A	2.9W	8.72mW	240mW	26.4mW	5.4W
P_A, MOPS/mm^2	343	3800	94	512	205.2	820.8
P_E, GOPS/W	N/A	180	24.4	85.3	38	29

ASPA chip represents experimental data, whereas the second represents expected results for 300 MHz clock. As it can be seen from the Table II, the ASPA yields to some new architectures [25], [29], [30], which are built in finer CMOS technologies, but is comparable with other solutions and competes in terms of performance and area efficiency with the devises fabricated in 0.18 μm process [26], [27]. However, it should be emphasized that despite absolute performance figure advantage the actual performance in low-level operations is comparable with ASPA chip. The application domain for the majority of referenced in Table II designs is low- and medium-level image processing. Yet from the published results [25], [26], [30], it can be observed that already for 128 $\times$ 128 images spatial and temporal filtering operations (e.g., contrast detection, convolutions, median filtering) takes less time when executed on ASPA chip.

The result of comparison of the fabricated chip and the expected characteristics of the scaled 128 $\times$ 128 ASPA chip with its closest digital and analog counterparts are summarized in Table III. The overall performance is estimated for grayscale operations, by which we imply arithmetic operations (excluding multiplication) of 8-bit precision. It can be observed that the ASPA chip compares favorably with other chips in terms of performance figures. The larger numbers for the ACE16k chip relate to performance estimation based on CNN-type algorithms [31]. In terms of conventional image processing these designs are more evenly matched. For example smoothing and Sobel edge detection requires 4 μs on ACE16k and 2.7 μs and 5.5 μs on ASPA (75 MHz).

The power efficiency still yields to analog designs; however, this parameter will improve with adjusting and scaling this design to a finer CMOS technology. Finally, the overall ASPA performance compared to other vision chips [5], [7], [8] is significantly boosted by the gain from asynchronous operations. With 0.76 ns propagation delay per pixel the performance of 128 $\times$ 128 array during asynchronous processing is estimated to be equivalent to over 100 TOPS on a synchronous SIMD array. Of course, the actual performance in real applications is lower due to the use of local synchronous operations; however, the

fact that the global operation takes only a few clock cycles to be accomplished reduces the processing time and power consumption significantly.

VI. Conclusions

We have presented a general-purpose vision chip with real-time focal plane grayscale processing capabilities. The main benefit of the presented architecture can be found in applications where both high performance and low power consumption are of paramount importance, while limited image resolution is acceptable. It is suitable for real-time computer vision applications such as surveillance cameras, autonomous robots, industrial machine vision, interactive toys and augmented vision applications.

The 22 $\times$ 19 proof-of-concept ASPA chip was fabricated in a 0.35 μm technology. It has been tested and is fully functional. The architecture, based on cellular processor array, is scalable to larger dimensions and finer technologies. By combining synchronous and asynchronous operations, this design attempts to extend the functional capabilities of smart sensors from low- to medium-level processing. Operating at 2.5 V and 75 MHz, the chip delivers 15.6 GOPS for binary and 1 GOPS for grayscale operations and compares favorably with digital and analog architectures that are based on a similar processor-per-pixel approach. Due to relatively large cell size of $100\times117\ \mu\text{m}^2$ current design (in 0.35 μm technology) is not economically expedient to fabricate as a high resolution device (e.g., VGA resolution chip would result in 7 $\times$ 6 cm silicon die). Additionally, the proposed device falls behind analog designs in terms of power efficiency. However, a fully digital approach promises simple adaptation to much finer technology, resulting in small pitch and reduced power consumption. Moreover, our layout designs suggest that by introducing two more routing layers the cell area is reduced by 30%. To a considerable degree this work was motivated by current developments in 3-D IC technologies, which enable accommodation of various parts of the design on separate layers (tiers) depending on their functionality [32], e.g., photo-sensing, analog processing, digital logic, memory, etc. Each of these layers can be implemented on technology best suited for its purposes. Hence, processing part can be fabricated in sub-100 nm technology, while sensing and analog parts in a more conservative process technology.

References

[1] K. E. Batcher, "Design of a massively parallel processor," *IEEE Trans. Comput.*, vol. c-29, no. 9, pp. 836–840, 1980.

[2] M. J. B. Duff and D. M. Watson, "The cellular logic array image processor," *Comput. J.*, vol. 20, no. 1, pp. 68–72, 1977.

[3] A. Moini, *Vision Chips or Seeing Silicon*. Norwell, MA: Kluwer, 1999.

[4] K. Aizawa, "Computational sensors-vision VLSI," *EICE Trans. Inf. Syst.*, vol. E82-D, no. 3, pp. 580–588, 1999.

[5] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analogue SIMD vision chip," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 52, no. 1, pp. 13–20, 2005.

[6] J.-E. Eklund, C. Svensson, and A. Astrom, "VLSI Implementation of a focal plane image processor—A realization of the near-sensor image processing concept," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 3, pp. 322–335, 1996.

[7] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 191–199, 2003.

[8] G. C. Linan, A. Rodriguez-Vazquez, and R. C. Galan *et al.*, "A 1000 FPS at 128 $\times$ 128 vision processor with 8-bit digitized I/O," *IEEE J. Solid-State Circuits*, vol. 39, no. 7, pp. 1044–1055, 2004.

[9] F. Paillet, D. Mercier, and T. M. Bernard, "Second generation programmable artificial retina," in *Proc. 12th Annu. IEEE Int. ASIC/SOC Conf.*, 1999, pp. 304–309.

[10] J. Poikonen, M. Laiho, and A. Paasio, "MIPA4k: A 64 $\times$ 64 cell mixed-mode image processor array," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taiwan, 2009, pp. 1927–1930.

[11] M. Wei, L. Qingyu, Z. Wancheng, and W. Nan-Jian, "A programmable SIMD vision chip for real-time vision applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1470–1479, 2008.

[12] B. Galilee, F. Mamalet, M. Renaudin, and P. Y. Coulon, "Parallel asynchronous watershed algorithm-architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 1, pp. 44–56, 2007.

[13] M. Habibi and M. Sayedi, "Geometric centre tracking of tagged objects using a low power demodulation smart vision sensor," *IET Circuits, Devices & Syst.*, vol. 4, no. 1, pp. 67–77, 2009.

[14] T. Komuro, A. Iwashita, and M. Ishikawa, "A QVGA-size pixel-parallel image processor for 1,000-fps vision," *IEEE Micro*, vol. 29, no. 6, pp. 58–67, 2009.

[15] N. Takahashi, K. Fujita, and T. Shibata, "A pixel-parallel self-similitude processing for multiple-resolution edge-filtering analog image sensors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 11, pp. 2384–2392, 2009.

[16] P. Nadrag, A. Manzanera, and N. Burrus, "Smart retina as a contour-based visual interface," presented at the Distributed Smart Cameras Workshop (DSC), Boulder, CO, 2006.

[17] A. Lopich and P. Dudek, "ASPA: Focal plane digital processor array with asynchronous processing capabilities," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2008, pp. 1592–1595.

[18] A. Lopich and P. Dudek, "A processing element for a digital asynchronous/synchronous vision chip," in *Proc. IASTED Int. Conf. Circuits, Signals, Syst.*, 2006, pp. 296–301.

[19] A. Lopich and P. Dudek, "Asynchronous cellular logic network as a co-processor for a general-purpose massively parallel array," *Int. J. Circuit Theory Applications*, 2010, to be published.

[20] P. Dudek, "A flexible global readout architecture for an analogue SIMD vision chip," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2003, pp. 782–785.

[21] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip I&II," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 657–666, 2004.

[22] A. Kitchen, A. Bermak, and A. Bouzerdoum, "A digital pixel sensor array with programmable dynamic range," *IEEE Trans. Electron Devices*, vol. 52, no. 12, pp. 2591–2601, 2005.

[23] A. Lopich and P. Dudek, "Global operations in SIMD cellular processor arrays employing functional asynchronism," in *Proc. IEEE Int. Workshop Comput. Archit. Mach. Perception Sens.*, Montreal, QC, Canada, 2007, pp. 16–23.

[24] Texas Instruments DSP TMS320C6455-1000 [Online]. Available: http://focus.ti.com/lit/ds/symlink/tms320c6455.pdf 2007

[25] A. A. Abbo, R. P. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, B. Vermeulen, and M. Heijligers, "Xetal-II: A 107 GOPS, 600 mW massively parallel processor for video scene analysis," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 192–201, 2008.

[26] P. Foldesy, A. Zarandy, C. Rekeczky, and T. Roska, "High performance processor array for image processing," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 1177–1180.

[27] S. Kyo, T. Koga, S. Okazaki, R. Uchida, S. Yoshimoto, and I. Kuroda, "A 51.2 GOPS scalable video recognition processor for intelligent cruise control based on a linear array of 128 4-way VLIW processing elements," in *IEEE Int. Solid-State Circuits Conf.*, 2003, pp. 48–477.

[28] L. Lindgren, J. Melander, R. Johansson, and B. Moller, "A multiresolution 100-GOPS 4-Gpixels/s programmable smart vision sensor for multisense imaging," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1350–1359, 2005.

[29] H. Noda, M. Nakajima, K. Dosaka, K. Nakata, M. Higashida, O. Yamamoto, K. Mizumoto, T. Tanizaki, T. Gyohten, Y. Okuno, H. Kondo, Y. Shimazu, K. Arimoto, K. Saito, and T. Shimizu, "The design and implementation of the massively parallel processor based on the matrix architecture," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 183–192, 2007.

[30] C. Chih-Chi, L. Chia-Hua, L. Chung-Te, and C. Liang-Gee, "iVisual: An intelligent visual sensor SoC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 127–135, 2009.

[31] P. Dudek, "Accuracy and efficiency of grey-level image filtering on VLSI cellular processor arrays," in *Proc. IEEE Int. Workshop Cellular Nanoscale Netw.*, 2004, pp. 123–128.

[32] P. Dudek, A. Lopich, and V. Gruev, "A pixel-parallel cellular processor array in a stacked three-layer 3D silicon-on-insulator technology," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, 2009, pp. 193–19.

Alexey Lopich received the M. Eng. degree in mathematical electronics from the Faculty of Mathematics and Mechanics at Belarusian State University, Minsk, Belarus, in 2003 and the Ph.D. degree from University of Manchester, Manchester, U.K., in 2007.

He has worked as a microelectronic engineer in 2003 and has been working as a Research Associate at the University of Manchester since 2006. His research interests are in image processing architectures and algorithms, VLSI design, smart sensors, machine vision, and embedded systems.

Piotr Dudek received the Mgr. Inz. degree in electronic engineering from the Technical University of Gdańsk, Poland, in 1997 and the M.Sc. and Ph.D. degrees from the University of Manchester, U.K., in 1996 and 2000, respectively.

He was a Research Associate and Lecturer in Integrated Circuit Engineering at the University of Manchester Institute of Science and Technology (UMIST), and currently he is a Senior Lecturer in the School of Electrical and Electronic Engineering, University of Manchester, leading the Microelectronics Design Lab. In 2009 he was a Visiting Associate Professor at the Hong Kong Univeristy of Science and Technoilogy (HKUST). His research interests are in analog and mixed-mode VLSI circuits, smart sensors, machine vision, cellular processor arrays and brain-inspired circuits and systems.

Dr. Dudek is a member of the IEEE Circuits and Systems Society Technical Committees on Neural Systems and Cellular Neural Networks and Array Processing, Secretary-Elect of the CAS Sensory Systems Technical Committee, and scientific committee member of several international conferences.