A PROCESSING ELEMENT FOR A DIGITAL ASYNCHRONOUS/SYNCHRONOUS VISION CHIP

Alexey Lopich and Piotr Dudek
School of Electrical and Electronic Engineering
The University of Manchester, UK;
E-mail: a.lopich@postgrad.manchester.ac.uk
p.dudek@manchester.ac.uk

ABSTRACT

This paper presents a new processing cell circuit, suitable for use in massively parallel fine-grain processor arrays, oriented towards image processing applications. The design, based on dynamic logic, is efficient for both local and global operations. In this paper we discuss design trade-offs and provide detailed description of the architecture. A cellular processor array based on the presented design can operate in both discrete- and continuous-time domains. Asynchronous execution of global operations significantly increases overall performance. Simulation results indicate the performance in the range from 1.1 (unsigned products) to 2900 (asynchronous binary processing) MOPS/cell.

KEY WORDS

Vision chips, cellular processor arrays, asynchronous processing, SIMD arrays, wave propagations.

I. INTRODUCTION

Early vision algorithms exhibit a high degree of parallelism and data regularity. Typical image pre-processing involves execution of identical instructions for each pixel. Massively parallel fine-grain processor arrays have been proven to be efficient for these applications, thanks to a pixel-per-processor approach. Employing single instruction multiple date (SIMD) paradigm, every processor associated with a pixel executes identical instructions issued by the central controller. In an attempt to design powerful and effective image processing systems, researchers often combine processing and sensing on a single silicon die, in a so-called 'vision chip'. This approach overcomes the bottleneck associated with transferring and processing a large amount of image data.

Cellular Processor Arrays (CPAs) with massively parallel architecture impose strict area constraints on processing elements (PEs). Smaller area of a PE results in the increased number of PEs that can be fitted on a chip. This fact has led to the popularity of analogue mode processing techniques for design of CPAs for image processing applications. A number of designs that operate in analogue mode have been presented in the literature [1-3]. The main advantage of this approach is the possibility of implementing greyscale operations while keeping the circuit area relatively small. Despite being favourable from the design size perspective, analogue mode has its own disadvantages, such as stability, noise, accuracy and error accumulation effect [4].

At the same time, continuous improvement in CMOS process technology opens new possibilities for building efficient digital designs. Several digital architectures of pixel-parallel processors and vision chips have been presented [5-7]. Because of limited memory capacity these solutions are suitable for binary image processing, with a possibility to be reconfigured for greyscale image processing to the prejudice of processing speed. Conventional digital design techniques are not optimised for such applications due to strict area constraints, therefore some alternative approach is required.

In this paper we present a novel VLSI architecture for a processing cell, that can be used in a general purpose CPA oriented towards image processing [8]. The design is based on dynamic memory and dynamic logic. Such an approach allows the size reduction of a PE and at the same time enables flexible functionality, low power consumption, programmability and an ability to reconfigure a number of PEs into a single network. First, design trade-offs, requirements and specifications are presented. Then, the architecture and circuitry are described. Next, the communication issues and asynchronous operation across the array will be discussed, followed by the design summary and conclusions.

II. DESIGN REQUIREMENTS

When designing circuitry for a PE, it is necessary to find an optimal trade-off between processing performance, area, speed and power consumption. During the early stages of vision chips research the main target was to satisfy strict area constraints even at the cost of processing performance and functionality [9]. Nowadays, the situation has changed because of several reasons. The higher level of integration allows fitting much more transistors per unit area. Emerging technologies will allow a three-dimension integration process, increasing packing density [10]. Finally, there is a growing tendency to use smart sensors for tasks of higher complexity that exceed the scope of image pre-processing [11]. These tasks include segmentation, object tracking, recognition and other global operations where both data parallelism and global data-flow across the pixel array are involved.

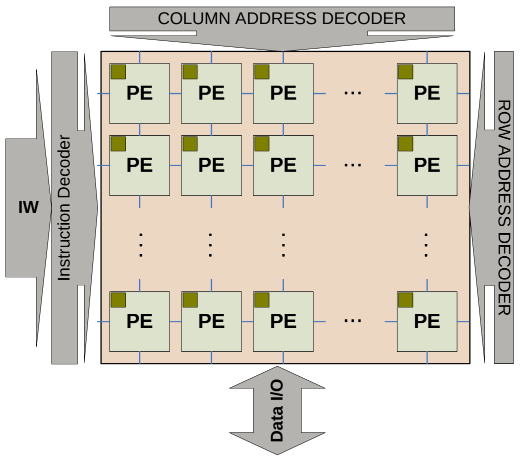

Figure 1: ASPA schematic diagram

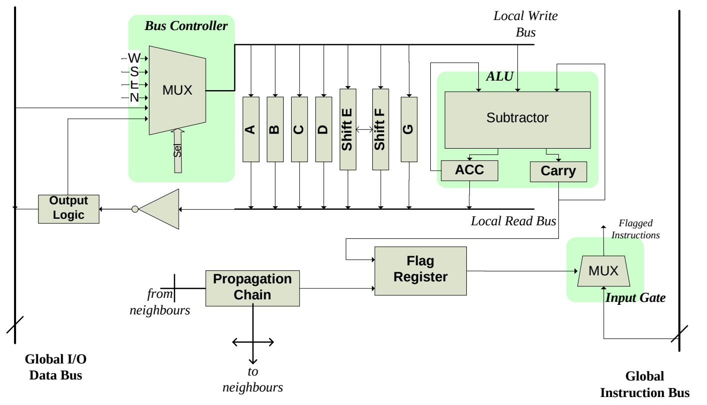

Figure 2: ASPA schematic diagram

The main goal of this work was to achieve an advanced functionality (in addition to local operations) of the processing cell, while keeping the PE size small enough to enable integration of up to 10^5 PEs on a single chip. The further goal was to enable the processor array to operate at high speed while keeping the power consumption low. We propose a digital architecture, based on dynamic memory and domino logic. Although dynamic logic has a few disadvantages (charge sharing, coupling capacitance), it significantly reduces design size (fewer transistors, dominance of one type of transistors). Considering the specifics of vision applications, such as frequent data refreshment, the store charge leakage can be either neglected or handled by periodic data reloading. Finally, while having a synchronous digital architecture, the PE is capable of executing certain operations in an asynchronous manner, thus achieving a significant performance increase when performing global operations.

Performance of a CPA always depends on functional capabilities of each cell. Initially, image pre-processing has always been a prime application for such devices. Therefore, processing cells were mainly oriented for efficient execution of local operators, i.e. a function that depends on information from the local neighbourhood and the pixel itself. The presented design extends pixel functionality by introducing an original approach to the execution of global instructions. Such operations involve data-flow across the array or distant pixel communications. To enable this feature we built a mechanism that avoids multiple iterations while executing global instructions. In addition to performance increase, asynchronous processing also exhibits reduced power consumption. From the application perspective the architecture benefits from the flexible programmability, so a wide range of image processing applications can be implemented in an efficient way.

For a feasibility study, we have designed the chip in a 0.35 μm CMOS technology, and all simulation results correspond to this technology. However, having a pure digital architecture we are able to scale all the design to a finer technology (e.g. 0.13 μm or 90 nm) with little redesign. Of course, finer technology may impose certain restrictions on design (e.g. due to higher leakage currents) and may require slight modifications, however the basic architecture and circuitry will remain the same.

III. Architecture

The PE is a basic building block of the asynchronous/synchronous processor array (ASPA) (Figure 1). Every PE is connected to its four neighbours so that data exchange is possible in four directions. The ASPA follows the SIMD paradigm, i.e. every PE is controlled by the same Instruction Word (IW). It is also possible to reconfigure the array into a single combinatorial circuit thus removing restriction of 'nearest neighbourhood' communication and enabling global asynchronous operations.

The structure of a PE is depicted in Figure 2. Essentially, the PE operates as a datapath with register-transfer operations controlled by a global program.

The PE contains 7 general purpose registers (GPR), arithmetic and logic unit (ALU), flag register, local bus controller and the propagation chain. Standard synchronous operation of a cell consists of reading the data from the local memory or ALU to the local read bus (LRB), forwarding it through the bus controller (BC) to the local write bus (LWB) and loading the data from the LWB to the local memory or ALU. At the same time, PEs can be configured in such a way that the whole ASPA will behave as a single combinatorial circuit, thus operating asynchronously.

Every PE employs mixed bit-serial and bit-parallel data processing. All register-transfer operations are performed in bit-parallel manner. In this way we increase the throughput of pixel communication and enable data manipulation during a transfer. However, all arithmetic calculations are performed in bit-serial manner. On the one hand, such an approach eliminates the dependence of the ALU size on the width of the processed data. On the other hand, we minimize the logic size and effects related to the propagation of the 'carry' signal.

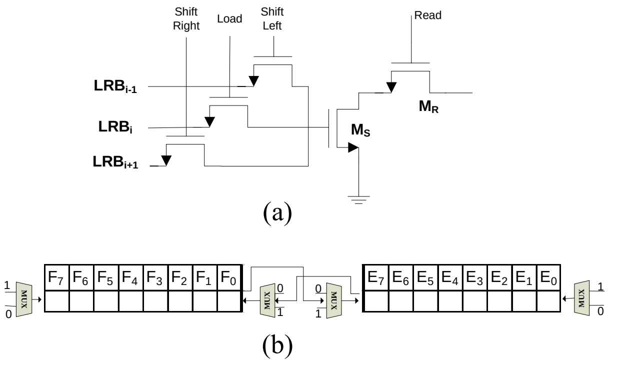

Figure 3: Shift Registers: a) Memory Cell; b) Connection of two shift reisters

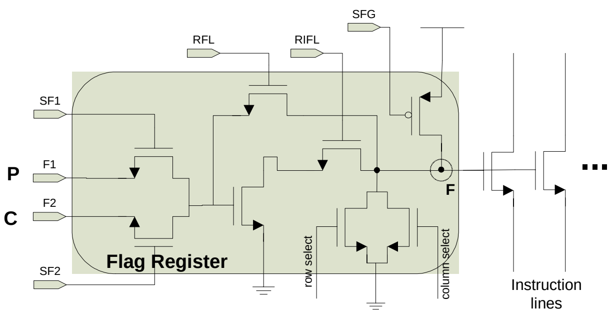

Figure 4: Flag Register

A. Local Memory

Memory capacity per cell is one of the most critical issues which influence the overall functionality and capability to implement different algorithms. When memory grows the PE area also increases. Therefore it is important to resolve this trade-off in order to provide the desired functionality in a small area. In the presented design we have implemented seven 8-bit digital registers. In addition to that, an accumulator register from the ALU can also be used as 8-bit memory storage. Thus, every PE is capable of storing 64 bits of information, which is a sufficient amount of memory to execute many low- and mid-level image processing algorithms. It should be emphasized, that it is possible to store and process several values shorter than 8-bit in the same register (e.g. two 4-bit values or use a register to store various 1-bit flags). This feature provides flexibility of memory usage for bit-serial and bit-parallel processing.

In order to comply with strict area constraints we employed a dynamic latch as a basic memory unit, which consists of three minimum-size transistors. The output of each cell is connected to the pre-charged LRB. Logic levels on the LRB are sensed by inverters that drive inputs of the BC. The leakage of the stored charge due to the reverse-bias and sub-threshold leakage currents will eventually corrupt the stored value, however, memory retention time is much less than the instruction rate and, if required, data refreshment can be performed. The latter procedure requires only two clock cycles per byte.

It can be noticed, that the reading of several memory elements to the LRB simultaneously will be equivalent to a NOR operation, i.e. additional logic operations can be performed while reading the values.

In addition to standard registers, we have implemented two bi-directional shift registers. Based on modified memory cell (Figure 3(a)) these registers allow immediate operand shift in both directions with indication of carry-in signal.

Memory cells, assigned to store most and least significant bits, are modified to allow dealing with contents of these two registers as with a single 16-bit datum, i.e. it is possible to shift data in two registers separately, as well as simultaneously so that upper bit of register **E** is loaded into lower bit of the register **F**, and vice versa (Figure 3(b)). Shift operation also enables the efficient execution of global asynchronous distance transforms and unsigned multiplication and division.

B. Conditional Operation

Enabling conditional branching within every cell introduces a locally autonomous operation. In the presented design, it is implemented by introducing an activity flag mechanism. The schematic diagram of the Flag Register (FR) is shown in Figure 4. Some of the instruction lines contain pass transistors that are controlled by an activity flag value. Thus, if the PE has an activity flag set to logic ‘0’, it will ignore instructions broadcast through flag-controlled lines.

The condition selection is performed by selecting the appropriate flag value pulling SF1 or SF2 high, so that the corresponding value is transferred to a storage node. The F1 input is connected to the propagation indicator **P** and F2 corresponds to a carry signal **C**. To enable branching on both conditions, actual and complement (e.g. *IF(C)* and *IF(!C)*), a special flag reading scheme is introduced. During unconditional operation the value SFG is logic ‘0’ and both Read FLag (RFL) and Read Inverted FLag (RIFL) instruction bits are set to logic ‘0’. Pass transistors on instruction lines are ON and the PE executes all transmitted instructions. If a conditional instruction such as *IF (C)* is met, then SF2, RFL and SFG are set to ‘1’, so that value **C** appears at node **F**. If the conditional check is performed on a complement flag value, as in *IF(!C)* , then the RIFL bit has to be set to logic ‘1’ instead of RFL. The flag register operates as a dynamic memory latch described above. When executing conditional instructions, it is important to set the instruction word to its initial state before reading the flag value.

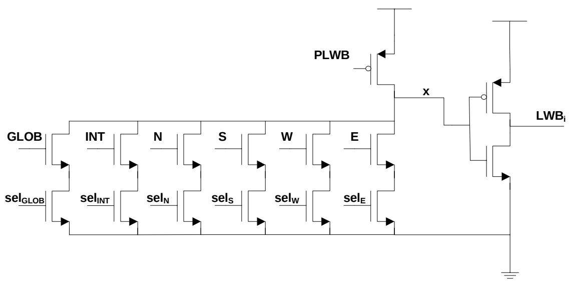

Figure 5: Bus Controller

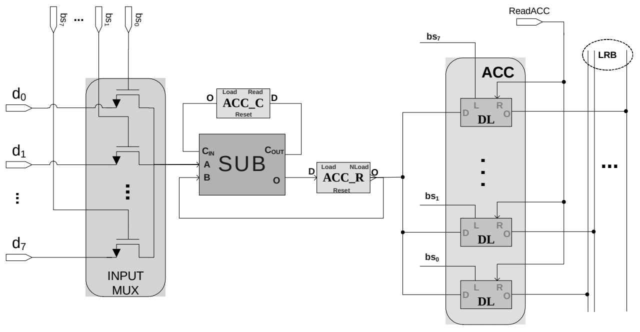

Figure 6: Bit-serial ALU

As stated above, the PE operates as a datapath of a microprocessor based on data-transfer operations. Exploiting this fact, we have reduced design size by applying conditional execution only to register load operations (LA, LB, LC, etc.). In addition to that, the flag register also controls two propagation-control signals (to set initial propagation centres and define the propagation space) and one register-read operation (to enable division operations and flexible data transfers). From an array operation perspective, this implies that those PEs that do not satisfy the condition will still perform instructions related to memory reading and arithmetic calculations, but all the "store" instructions will be ignored. They will remain "disabled" until an *ENDIF* instruction is met (SFG bit is set to '0').

C. Bus Controller

The bus controller (Figure 5) is associated with multiplexing internal and external data into the LWB. From the circuit design perspective it is a dynamic OR gate. Control signals indicate what value is to be transferred to the LWB, thus enabling the following set of operations: GPR → GPR (ALU), Global I/O Bus → GPR (ALU), Neighbour → GPR (ALU). The process of data transfer consists of two phases: 1) Node ***x*** is charged with V_{DD}, all the select signals are '0'; 2) The data is selected by appropriate signals and is outputted through the inverter to LWB. A logical OR operation within BC facilitates global data-transform operations.

D. I/O interface

The data readout procedure is akin to the local memory readout. The LRB of the PE is connected to a global column output bus when the row select signal is set to '1', thus performing column parallel readout. The peripheral circuitry will multiplex data to a global output according to column select signals. The value of the flag register can also be read-out in an identical manner. Reading-out binary data (flag values) can significantly increase system throughput (8 values can be simultaneously read-out in parallel) and be used to perform a global OR operation.

Global input data is delivered using the same bus as for read-out. To ensure correct operation, it is necessary to pre-charge the LRB each time the global data is broadcast.

E. ALU

The ALU is assigned to perform arithmetic and logic operations within each cell. To satisfy strict area constraints, we employed a bit-serial architecture, thus reducing the size of the design.

The presented ALU is able to perform logical XOR operation and unsigned arithmetic subtraction. The block diagram of the ALU is presented in Figure 6. 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 the accumulator for the final result (ACC).

The subtractor unit is based on a 'pass' logic design and consists of only 12 minimum size transistors. Its outputs are stored in two dynamic flip-flops (ACC_R and ACC_C). Since the output of the subtractor is degraded because of the 'pass' logic, careful transistor sizing is required for the front inverter of these flip-flops.

The input multiplexer, controlled by signals $bs_0, bs_1, \ldots, bs_7$, is intended to select the processing bit. We use a 'one hot' coding method for bit selection. Since all data lines ($LWB_0, LWB_1, \ldots, LWB_7$) are driven by designated drivers in BC, applying '1' to several bit select inputs may lead to signal conflicts**,** and therefore it is a prohibited combination.

As a functional block, the SUB executes the following operation:

$$O = A \oplus B \oplus C_{IN} \quad (1)$$

$$C_{OUT} = \bar{A}B \oplus \bar{A}C_{IN} \oplus BC_{IN} \quad (2)$$

In order to subtract two 8-bit numbers, e.g. D=A-B, the following steps have to be repeated: 1) Reset ACC_R and ACC_C (reset implies that the output is set to '0' but the stored value is preserved), select the appropriate bit; 2) Read the register B and load the result to ACC_R (ACC_R=b_i-0); 3) Pre-charge the LRB and the LWB, read the value from ACC_R (b_i) and ACC_C (c_{i-1}). 4) Read the register A and load the value to ACC_C (c_i) and ACC_R (d_i).

Thus, during four clock cycles the following values are calculated:

$$d_k = a_k \oplus b_k \oplus c_{k-1} \quad (3)$$

$$c_k = \bar{a}_k b_k \oplus \bar{a}_k c_{k-1} \oplus b_k c_{k-1} \quad (4)$$

It should be emphasized that signals bs_i (i=[0,7]) are also used as 'load' controls for the accumulator ACC.

Availability of two bi-directional shift registers provides a flexible tool for unsigned shift-and-add multiplication and shift-and-subtract division. These two operations are extensively used in discrete linear transforms executed directly on the ASPA [8].

IV. Pixel Communication

Overall CPA performance significantly depends on how efficient pixel-to-pixel communication is. This dependency is especially important for global operations, when the data-flow range exceeds the local neighbourhood. The number of iterations required for synchronous data transfer exhibits quadratic dependence on the array size. Hence, asynchronous implementation of this operation can significantly speed up the process. In fact, the peak speed will be limited by the signal propagation delay through the PE circuitry. The power consumption is also optimised, because the number of active elements is decreased and is equal to the number of wave-front elements [12].

Most of the designs presented in the literature employ the SIMD paradigm and provide 'instant' communication only to nearest neighbours. Some of the presented architectures employ dynamic reconfiguration and PE chaining [3, 11]. One of our major design goals was to enable basic global asynchronous operations that can form a basis for implementation of algorithms involving inter-pixel dataflow. After analysing several image segmentation algorithms, we concluded that two most frequently used global operations are binary trigger-wave propagation and distance transform. More complex algorithms, such as skeletonization, watershed transform, object reconstruction and other morphological and image segmentation procedures can be decomposed into a number of simple global operations and local convolution operators.

Current design provides instant access to local neighbourhood values through the BC. It can be noted that communication through the BC makes the pixel-to-pixel data transfer procedure identical to the local register-register operation and requires two clock cycles per 8-bit transfer. In other words, every PE has an instant access to LRBs of its nearest neighbours. However, this also gives a possibility to chain PEs and perform cumulative operations (for example to calculate global summation within a defined block).

The main advantage of such an approach is the potential to distribute data asynchronously within a defined block of pixels, or between two distant pixels. This procedure can be effectively used in a variety of data reduction algorithms, when, for example, the average value has to be distributed within a block. Moreover, it is possible to transfer and process data at the same time, so that algorithms such as the distance transform can be accomplished in an asynchronous manner.

One of the examples of processing data while transferring asynchronously is incrementation. It is possible to program a PE so that it will perform the following operations simultaneously: read the data from all the neighbours, shift the obtained data and load it into dedicated register, read the data into LRB, thus making it accessible by other neighbours. By representing the data in appropriate way we can achieve that OR operation in the BC will calculate the minimum among the neighbours and shifting will correspond to an increment operation. In this way the exact Manhattan distance can be calculated within eight pixel distance range [8].

In general, it may take longer than a single clock cycle to distribute data across the array, because of the propagation delay introduced by PEs. The achieved acceleration $\boldsymbol{A}$ can be calculated as follows:

$$A = \frac{n_I \cdot t_I}{t_d} \tag{5}$$

Where n_I is the number of instructions, necessary to pass the value in a PE to its local neighbour, t_I is the duration of instruction cycle, i.e. time to deliver and execute instruction in all PEs, t_d is the propagation delay per PE. The instruction supply time is always greater than t_d, because the maximum working frequency is limited by the propagation delay inside the circuitry of every PE. Moreover, asynchronous operation results in reduced power consumption due to fewer PEs being involved in actual processing.

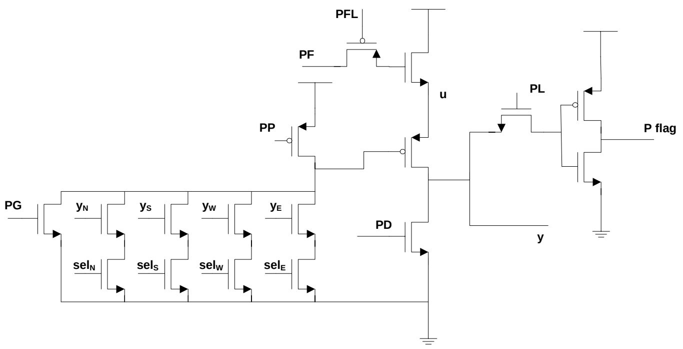

Figure 7: Propagation Chain

To extend the flexibility of asynchronous processing we have implemented an additional functional block – propagation chain (Figure 7). This unit provides a simple tool for binary trigger-wave propagations across the pixel array that can be used to perform global operations such as object reconstruction and hole filling in a fast and power-efficient way. The more detailed description of the circuit can be found in [13] and here just a brief introduction is provided.

The initial binary image is supplied to the input PFL and then conditionally drives node u. Initial state is loaded to PG. Values y_N, y_S, y_W and y_E represent the present state of the nearest neighbours, whereas sel_N, sel_S, sel_W and sel_E are masking signals. The output function will look as follows:

$$y = (y_N \cdot sel_N + y_S \cdot sel_S + y_W \cdot sel_W + y_E \cdot sel_E) \cdot u + u \cdot PG \tag{6}$$

The propagation across the array resembles a domino effect and is carried out in an asynchronous manner with sub-nanosecond per PE signal propagation delay. Masking signals correspond to four least significant values of the LRB, i.e. $sel_N = LRB_3$, $sel_W = LRB_2$, $sel_S = LRB_1$, $sel_E = LRB_0$. Thus, the direction of propagation is controlled from within each individual PE.

V. Implementation Summary

We have implemented 19x22 array chip in a 0.35 µm CMOS technology. The PE circuitry consists of 450 minimum size transistors. 94% of them are n-type transistors. The area of each PE is 100µm x 117µm (Figure 8), thus achieving 85 cells/mm^2. Hence, a 128x128 array could be fabricated on a 200 mm^2 chip (including peripheral circuitry). The overall performance is estimated according to the executed operation (binary, greyscale, unsigned multiplication and division) and operation mode (synchronous, asynchronous). Performance estimation for asynchronous mode was performed by implementing the same algorithm in synchronous fashion and comparing the number of required operations with the number required for asynchronous realisation. Therefore, according to SPICE simulations, every PE in the ASPA provides 83 MOPS (binary), 10 MOPS (greyscale) and 1.1 MOPS (for unsigned products and quotient) in a synchronous mode. Execution of basic edge detection with a Sobel operator on the ASPA takes 1.3 µs. The achieved area usage parameter is 1.2 GOPS/mm^2 (for greyscale operations). Using lower-scale technology these figures can be optimised. The performance characteristics for 128x128 array are as follows: 1.36 TOPS (binary), 163.84 GOPS (greyscale), 18.02 GOPS (for unsigned products). SPICE simulations showed that operating at 300MHz clock the chip consumes 400 µW/cell providing 9x10^{-9} OP/Joule (greyscale). The signal propagation delay through a pixel was approximately 0.25 ns. Considering Eq. (6), the asynchronous implementation provides at least a 36 times speed increase compared to synchronous operation. Thus for binary operation in an asynchronous mode the performance per cell is estimated to be 2900 MOPS. The power consumption savings express similar dependency. In case of trigger-wave propagations on the ASPA, the worst case of energy consumption (i.e. all the PEs lay within the propagation space and are triggered) is 0.4 pJ per PE.

VI. CONCLUSIONS

We have presented a new general-purpose processing cell. A CPA based on the presented design is suited for versatile image processing applications of varied complexity. The design operates in digital mode, thus offering scalability and eliminating noise and accuracy problems of analogue approach. The circuit design is based on dynamic logic, providing 64 bits of memory storage per pixel. Unsigned multiplication and division are supported thanks to two shift registers available in each PE. Therefore various linear discrete image transformations can be efficiently executed in parallel mode. The presented design allows chaining PEs into a single network, thus facilitating distant data transfers, global feature extraction, binary trigger-wave propagations and other operations that involve data-flow across a pixel network.

The circuit has been simulated in SPICE and achieved results have been presented. The processor exhibits ultra high performance, flexible programmability together with reasonable size and power consumption.

VII. ACKNOLEGEMENT

This work has been supported by the EPSRC grant number EP/D029759. The authors would like to thank Dr. S.J.Carey for valuable remarks.

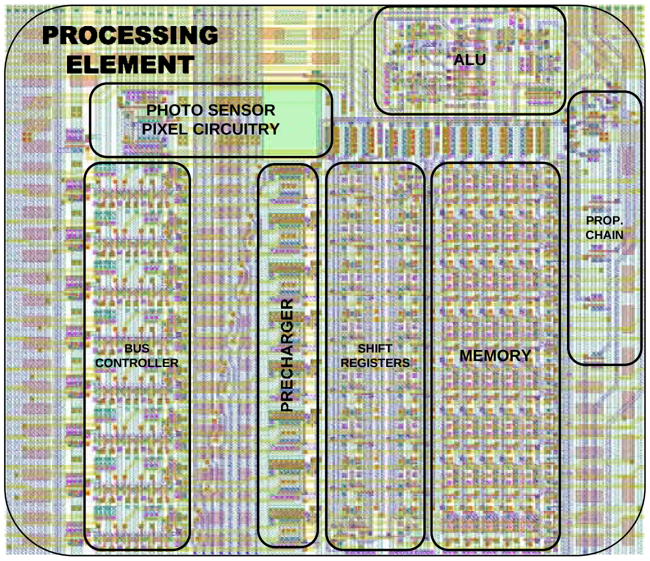

Figure 8. Processing Cell Layout

REFERENCES

[1] Linan, G.C., Rodriguez-Vazquez, A., *et al.*, *A 1000 FPS at 128 x128 vision processor with 8-bit digitized I/O.* IEEE Journal of Solid-State Circuits, 2003 European Solid State Cirquits Conference (ESSCIRC), 2004. **39**(7): p. 1044-1055.

[2] Dudek, P.,Hicks, P.J., *A general-purpose processor-per-pixel analog SIMD vision chip.* IEEE Tran.on Circuits and Systems I: Fundamental Theory and Applications, 2005.**52**(1):p.13-20.

[3] Paillet, F., Mercier, D., Bernard, T.M. *Second generation programmable artificial retina*. in *Twelfth Annual IEEE International ASIC/SOC Conference, 15-18 Sept. 1999*. 1999. Washington, DC, USA: IEEE. p.304-309

[4] Dudek, P. *Accuracy and Efficiency of Grey-level Image Filtering on VLSI Cellular Processor Arrays*. in *CNNA 2004*. 2004. Budapest. p.123-128

[5] Komuro, T., Ishii, I., *et al.*, *A digital vision chip specialized for high-speed target tracking.* IEEE Transactions on Electron Devices, 2003. **50**(1): p. 191-199.

[6] Eklund, J.-E., Svensson,C., Astrom, A., *VLSI Implementation of a Focal Plane Image Processor - A Realization of the Near-Sensor Image Processing Concept.* IEEE transaction on VLSI Systems, 1996. **4**(3): p. 322-335.

[7] Gealow, J.C.,Sodini, C.G., *Pixel-parallel image processor using logic pitch-matched to dynamic memory.* IEEE Journal of Solid-State Circuits, 1999. **34**(6): p. 831-839.

[8] Lopich, A., Dudek, P. *Architecture of a VLSI cellular processor array for synchronous/asynchronous image processing*. in *ISCAS2006*. 2006.

[9] Moini, A., *Vision Chips or Seeing Silicon*. 1999: Kluwer Academic Publishers

[10] *ITRS Roadmap.* http://public.itrs.net.

[11] Komuro, T., Kagami, S., Ishikawa, M., *A dynamically reconfigurable SIMD processor for a vision chip.* IEEE Journal of Solid-State Circuits, 2004. **39**(1): p. 265-268.

[12] Lopich, A., Dudek, P. *Architecture of asynchronous cellular processor array for image skeletonization*. in *ECCTD 05*. 2005. Cork, Ireland: IEEE. p.81-84

[13] Dudek, P., *An Asynchronous Cellular Logic Network for Trigger-Wave Image Processing on Fine-Grain Massively Parallel Arrays.* IEEE Transactions on Circuits and Systems II (accepted).IEEExplore:DOI 10.1109/TCSII.2006.869916.