Implementation of an Asynchronous Cellular Logic Network as a Co-Processor for a General-Purpose Massively Parallel Array

Alexey Lopich *Piotr Dudek*
School of Electrical & Electronic Engineering,
The University of Manchester,
PO Box 88, Manchester, M60 1QD, United Kingdom;
E-mail: {a.lopich, p.dudek}@manchester.ac.uk

Abstract — **In this paper we present an implementation of an asynchronous cellular processor array that facilitates binary trigger-wave propagations, extensively used in various image processing algorithms. The circuit operates in a continuous-time mode, achieving high operational performance and low power consumption. A 24×60 proof-of-concept array integrated circuit has been fabricated in a 0.35µm 3-metal CMOS process and tested. Occupying only 16×8 µm^2 the binary wave-propagation cell is used as a co-processor in a general-purpose processor-per-pixel array that is designed for focal-plane image processing. The results of global operations such as object reconstruction and hole filling are presented.**

I. INTRODUCTION

Due to a high degree of data-parallelism, low-level image processing algorithms can be efficiently implemented on processor arrays. In particular, the fine-grain parallelism of these algorithms is well suited to processor-per-pixel architectures [1-4]. These cellular processor arrays operate according to a single-instruction multiple data (SIMD) paradigm, which has been proven to be effective for operations where the pixel is explicitly expressed as a function of its bounded neighbourhood [5]. However, there are a number of low- and medium-level image processing operations (object reconstruction, distance transform, hole filling and other morphological and segmentation routines) that involve global data-flow across pixel network. These operations can be represented as an iterative process, where a pixel value, while determined as a function of local neighbourhood on each iteration, implicitly depends on the entire array of pixels. The meaningful processing during each iteration is typically carried-out only in a limited number of 'active' pixels, at a front of a propagating wave of activity (a flood-filling or a grass-fire algorithm are the classical examples of this process). When implemented on a fully parallel, synchronous processor-per-pixel SIMD array, such algorithm requires many iterative steps ($\mathbf{O}(N^2)$ for $N\times N$ image) executed concurrently on all $N\times N$ processors, and therefore it has a high computational cost (and correspondingly inefficient use of hardware and high power consumption). In order to optimise global operations on cellular processor arrays it is necessary to look for alternative to SIMD approaches [6-9]. In order to increase the speed and efficiency of global operation, one should either introduce a very complex network of global interconnections or optimise the flow through the network of locally interconnected cells. The first approach becomes very difficult for a physical realisation, since the dimensions of processed images (and correspondingly the complexity of interconnections in processor arrays) are continuously increasing, whereas the number of routing layers, available for interconnections, is limited. The second approach, based on local communication only, yields itself better to a VLSI implementation. In case of iterative wave-propagating algorithms a promising approach involves local triggering of processor cells by their neighbours, so that they only perform the required computations when the results of computations in their neighbouring cells become available. Since the timing of the entire operation is data-driven, asynchronous methodologies can be employed. Apart from speeding up the propagations, asynchronous execution provides savings in power consumption as only active "wave-front" pixel cells do perform the actual processing, whereas others remain in the idle mode.

Several asynchronous architectures for processor-per-pixel arrays have been presented in the literature [8, 10]. However, the algorithm-specific designs not only have a very limited functionality, but also occupy relatively large silicon area, which is the major drawback when used as a co-processor for general-purpose cellular processor array. Therefore, it is important to identify the global operation that is extensively used in computer vision and has a simple hardware realisation. Binary propagations can be efficiently employed for basic object reconstruction and hole filling [11] as well as a component of more complex algorithms such as watershed segmentation [12]. In order to explore the benefits of using an asynchronous sub-network within a general-purpose processor-per-pixel cellular array, we have implemented a general-purpose asynchronous/synchronous processor array (ASPA)[13]. We have also implemented an asynchronous cellular logic array (ACLA) [11] chip as a proof-of-concept design for evaluation of the actual performance of asynchronous processing. This paper reports the experimental results obtained with this implementation.

1-4244-1342-7/07/$25.00 ©2007 IEEE

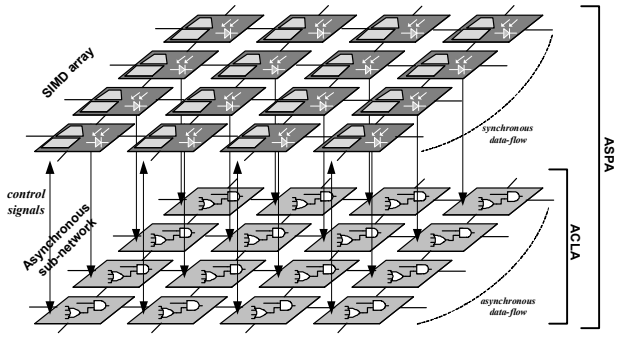


Figure 1: ASPA architecture – sub-networks with separate data-flows.

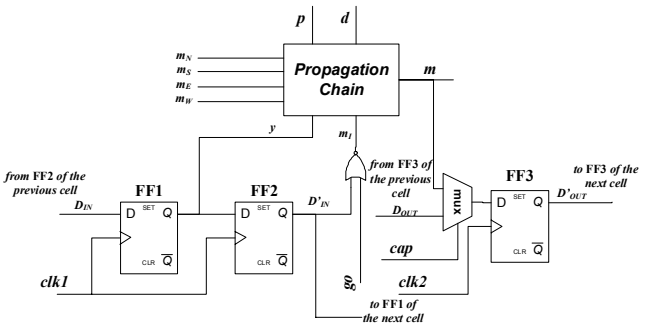


Figure 2: ACLA cell.

II. ASPA ARCHITECTURE

Asynchronous cellular logic networks, on its own, are not very useful for general-purpose image processing as they have fixed and limited functionality. Therefore, the ASPA vision chip [13] combines synchronous and asynchronous approaches, thus benefiting in terms of programmability and versatility while in the SIMD mode (discrete-time, synchronous operation) and achieving maximum performance in global operations when configured as a single combinatorial circuit (continuous-time, asynchronous operation).

The ASPA consists of an array of processing elements (PE), placed in a 4-connected rectangular grid. Every PE is a simple digital processor with arithmetic and logic unit (ALU) and general-purpose registers (GPRs). Eight 8-bit GPRs and an accumulator in the ALU enable every PE to store 64 bits of data. The PE operates as a datapath with register-transfer operations controlled by an instruction word. Operating in SIMD mode, all PE's execute the same instruction broadcast by a central controller. Local autonomy is provided by *carry* and *propagation* flag values.

Simple synchronous digital architecture of the cell facilitates all arithmetic and logic operations, thus, enabling efficient execution of most of the low-level image processing algorithms. In order to extend functional capabilities of the array from local to global operations, every cell is endowed with a simple functional unit, called Propagation Chain (PC), which provides a simple tool for binary trigger-wave propagations across the entire array, controlled by flag and mask indicators. Structurally, these asynchronous units can be considered as a separate sub-network (Figure 1) and this asynchronous network has been implemented as a separate chip for detailed evaluation of performance and power characteristics.

It should be noted, that the ASPA chip contains further mechanisms for asynchronous data transfers and operations as outlined in [13], however, discussion of these is outside the scope of this paper.

III. ACLA – OPERATION PRINCIPLES

The operation of every cell in the asynchronous sub-network is based on a Boolean function that is fundamental to geodesic reconstruction. Let us consider a 4-connected rectangular grid of cells and let us assign two Boolean values y and m to each cell. The value m corresponds to the 'marker' of the pixel ('1' – pixel is marked, '0' - unmarked) and the value y – to the 'pixel value' of input image ('1' – foreground, '0' – background). The process of reconstruction is carried-out by iterative calculation in each cell of the following Boolean function:

$$m = (m_N \vee m_S \vee m_W \vee m_E \vee m_I)\, y \qquad (1)$$

Where m_I is the initial marker and m_N, m_S, m_W and m_E correspond to four nearest neighbours. At the end of this global operation all pixels of initially marked objects are marked.

In order to provide further control of the propagation, for example to extend the use of binary propagations to greyscale-based algorithms [12] it is possible to constrain interconnection locally in every cell, so that the final function is as follows:

$$m = (c_N m_N \vee c_S m_S \vee c_W m_W \vee c_E m_E \vee m_I)\, y \qquad (2)$$

where (c_N, c_S, c_W, c_E) is a locally stored mask. In this way we achieve reconfigurable topology, i.e. the direction of the propagation is controlled by local data. The usefulness of such operation for image processing architectures has been demonstrated in the literature [2, 11, 14].

IV. IMPLEMENTATION

We have fabricated a proof of concept chip that incorporates an array of cells, which execute operations described in previous section in an asynchronous manner. The array is organised according to a rectangular grid arrangement of 24×60 cells with each cell interconnected to its four neighbours. The architecture of each cell is presented in Figure 2.

The I/O for the array is organised using shift registers, so that the data is input (via FF1-FF2 chain) and output (via FF3 chain) in a serial manner. The operation principle of the propagation chain sub-circuit is fully described in [11] and here we give only a brief description.

The propagation chain cell is shown in Figure 3. The circuit is based on dynamic logic. Both input image and initial marker are stored locally and applied to the inputs ***y*** and $\boldsymbol{m_I}$ correspondingly. Initially, the entire asynchronous network is discharged by applying logic '1' to the input ***d*** and the storage node **X** is precharged in every cell by setting the input ***p*** into logic '0'. After all initial steps are taken, the propagation is initiated by setting the ***go*** signal

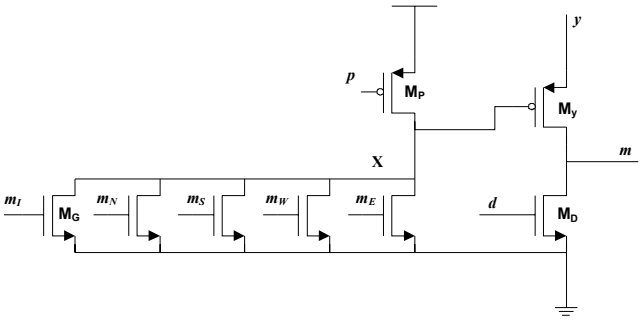


Figure 3: Schematic of the propagation chain.

(Figure 2) to logic '0' so that the inverted value of the initial marker is applied to the input m_I. By discharging the node **X** the value ***y*** is applied to the output ***m*** thus triggering four neighbouring cells, and so on. After the propagation is complete (or during the propagation, to capture the data for experimental purposes), the result is latched into the output flip-flop FF3. The input ***cap*** (Figure 2) is used as a select signal for a multiplexer, so that after the result is stored, the input of the flip-flop FF3 is connected to the output of the same flip-flop in the neighbouring cell to enable shifting-out the result.

The chip (Figure 4) is fabricated in a 0.35 µm 3-metal layer CMOS process (AMS). The size of the cell is 25×25 µm^2, of which the propagation chain cell comprises only 16×8 µm^2 (Figure 5). Such small size makes it attractive to incorporate this circuit as a co-processor to a general-purpose processing cell. In the ASPA chip [13] such asynchronous propagation unit occupies only 1% of the 100×117 µm^2 cell area.

V. RESULTS

Correct operation and performance of the ACLA chip have been verified by varying the delay (with resolution step of 5 ps) between the falling edge of the ***go*** signal (initiates the propagation) and the rising edge of the clock ***clk2*** (latches the state of the array). The output data showed that the average processing time per cell is approximately 0.48 ns. On an equivalent SIMD array (with one clock cycle per iteration) this time corresponds to 2.08 GHz clock. Interpolating this result to a 256×256 image, the frame processing time becomes 31.5 µs (the worst case, NxN iterations, although typically a smaller number could be used), which enables processing binary images at 31×10^4 images/sec and faster. Many propagations can be therefore performed within a single image frame, for example to implement multiple object reconstructions, or use propagations extensively as a component of more complex image processing operations [12]. The total energy consumed by triggered pixel during propagation is 0.4 pJ. Processing images at 31×10^4 frames per second the power consumption contribution of each cell is 12.7 nW, or 0.83 mW for a 256×256 array.

The performance of the ACLA chip could be favourably compared with arrays that provide similar facilities [2, 15]. The maximum throughput during asynchronous operation for the NSIP [2] chip is reported to be 4×10^6 frames per second, which corresponds to approximately 2ns delay per cell. In the case of the chip

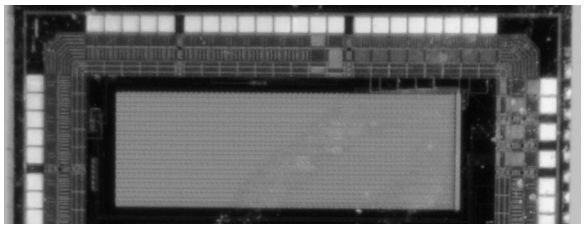


Figure 4: Microphotograph of the ACLA chip.

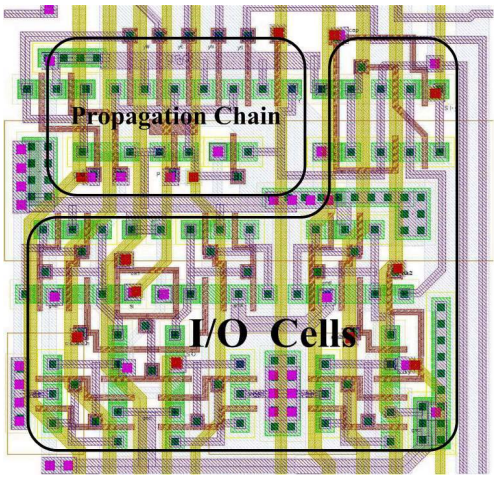


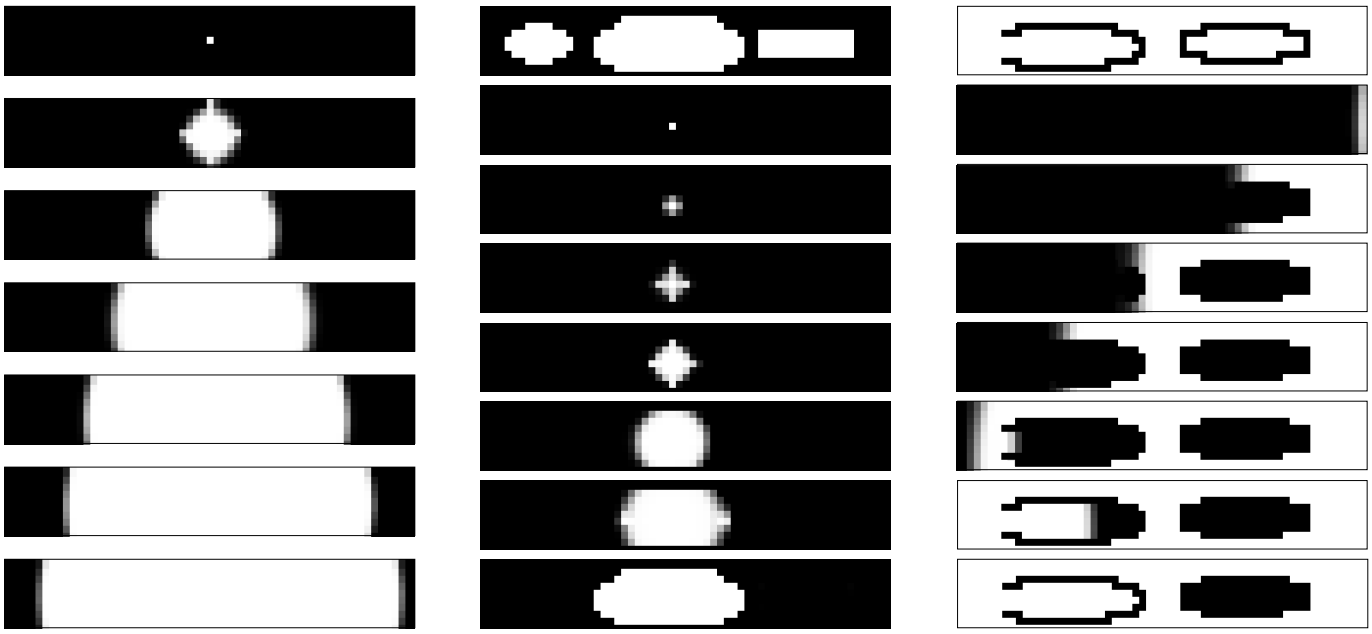
Figure 5: Cell layout

reported in [15], the propagation delay per cell is 4 ns (however this chip benefits from threshold capabilities and applies a 3×3 mask to the 8-connected neighbourhood in every pixel).

The results of processing are presented in Figure 6. To illustrate the operation of the circuit, the results were captured at various stages during the propagation. Multiple captures were performed, sweeping the time between the ***go*** signal and the ***clk2*** signal with a fine time-step, and the grey-level of a pixel in the frames plotted in Figure 6 corresponds to the number of 1's and 0's captured in that pixel, across all capture runs within the corresponding frame time. It is also interesting to notice, that the shape of the propagation front has a spherical rather than a diamond shape. The reason for this phenomenon is the increased speed of switching of the OR gate in the propagation chain circuit (Figure 3) when several neighbours are switching their states simultaneously. Due to some clock distribution issues not all the pixels in the array are accessible and therefore test images are smaller than the original size of the fabricated array. While transistor mismatch and parasitic capacitance may lead to slight differences in propagation speed across the array, the observed effect is very small and in either case, the result of the basic reconstruction/hole filling operation is not dependant on the propagation speed uniformity.

VI. CONCLUSIONS

An implementation of an asynchronous cellular array has been presented. The designed array combines small cell area, high processing speed and low power consumption. The performance and power characteristics of the circuit



a) Wave propagation (*step*=2ns) b) Object reconstruction (*step*=1.8ns) c) Hole filling (*step*=5ns)

Figure 6: Global operations, implemented on a 24×60 asynchronous cellular processor array.

have been evaluated and indicate that it can be efficiently used as a co-processor in a general-purpose processor-per-pixel array in order to enable global operations at minimum hardware cost. The size of asynchronous part of the circuit is 16×8 μm^2 and in the ASPA chip realisation it consumes only 1% of the overall cell area. Digital architecture of the asynchronous unit suggests that it is comfortably scalable to finer technologies. The on-going research on this topic will concentrate on the development of a new reconfigurable architecture of the general-purpose processing cell that would facilitate asynchronous processing utilising the same circuitry that is used during the synchronous operation.

REFERENCES

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

[2] 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.

[3] 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 Circuits Conference (ESSCIRC), 2004. **39**(7): p. 1044-1055.

[4] 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.

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

[6] Ducourthial, B.,Merigot, A., *Parallel asynchronous computations for image analysis.* Proceedings of the IEEE, 2002. **90**(7): p. 1218-1229.

[7] Galilee, B., Mamalet, F., Renaudin, M.,Coulon, P.Y., *Parallel asynchronous watershed algorithm-architecture.* IEEE Transactions on Parallel and Distributed Systems, 2007. **18**(1): p. 44-56.

[8] Robin, F., Renaudin, M.,Privat, G. *An asynchronous 16*16 pixel array-processor for morphological filtering of greyscale images*. in *Proceedings of ESSCIRC '96, 17-19 Sept. 1996*. 1996. Neuchatel, Switzerland: Editions Frontieres. p.188-191

[9] Manzanera, A. *Morphological segmentation on the programmable retina: towards mixed synchronous/asynchronous algorithms*. in *6th International Symposium on Mathematical Morphology*. 2002. Sydney, Australia. p.389-399

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

[11] 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: Analog and Digital Signal Processing, 2006. **53**(5): p. 354-358.

[12] Lopich, A., Dudek P. *Global operations in SIMD cellular processor arrays employing functional asynchronism*. in *CAMPS 2006*. 2006. Montreal. p.18-23

[13] Lopich, A., Dudek, P. *Architecture of a VLSI cellular processor array for synchronous/asynchronous image processing*. in *IEEE ISCAS*. 2006. p.3618-3621

[14] Manzanera, A., *Morphological Segmentation on The Programmable Retina: Towards Mixed Synchronous/Asynchronous Algorithms*. Proceedings of ISMM2002, 2002.

[15] Flak, J., Laiho, M., Paasio, A.,Halonen, K., *Dense CMOS implementation of a binary-programmable cellular neural network.* International Journal of Circuit Theory and Applications, 2006. **34**(4): p. 429-443.