# ARCHITECTURE OF ASYNCHRONOUS CELLULAR PROCESSOR ARRAY FOR IMAGE SKELETONIZATION

Aliaksei Lopich*  Piotr Dudek*

*Abstract* – **This paper presents a design and implementation of an application specific cellular processor array (CPA) that executes binary image skeletonization on a hexagonal lattice. The designed CPA operates in an asynchronous mode, employing 'pixel per processor' concept, which provides significant performance increase in image processing operations that exploit 'wave-propagation' or 'grassfire' transformation approach. A proof-of-concept design has been implemented and evaluated in FPGA and results are presented and discussed.**

## 1 INTRODUCTION

Cellular Processor Arrays (CPA's) have attracted significant attention in the last ten years, as they have shown the potential to achieve high performance, small area and low power consumption, as compared to conventional vision systems. Employing 'processor per pixel' approach, CPA's provide a solution for vision systems capable of performing various image processing operations at high speeds. Most of the CPA designs employ an SIMD paradigm and operate in a synchronous mode [1-3]. The SIMD concept is efficient in low-level image processing operations. However, there are a number of image processing algorithms, such as object reconstruction, hole-filling and skeletonization, which require a number of iterations and where the data-flow resembles a wave-propagation process. In this case parallel processing of only "wave-front" pixels is required, and thus SIMD approach will result in inefficient power consumption by the majority of the processors in the array. Asynchronous implementation of such global operations can provide up to $N^2$ performance increase on $N$x$N$ image. Apart from speeding up the execution process such approach is power efficient, because only the active pixels perform logic functions.

A hardware realisation of an asynchronous CPA's depends on the target algorithm. If the processing procedure consists of a single instruction then implementation of such an array is rather straightforward and employs only combinatorial logic [4]. A single-state asynchronous propagation networks are also represented by a near-sensor processing systems [5] and single-layer Cellular Neural Networks. However, if each processing element (PE) is to accomplish several iterative steps or the propagation process is data-dependent, then it is necessary to implement self-timed approach, which implies operation in an asynchronous mode according to the control signals, which are generated internally. Such local control strategy significantly extends the functional flexibility of a cell.

In this paper we present a method for skeleton extraction in a wave-propagating fashion, suitable for VLSI implementation. The designed CPA operates in an asynchronous mode which enables performing its function within a "single instruction", i.e. within one cycle of the global clock [6]. The result of the operation is a two-pixel wide skeleton extracted from the original binary image loaded into the array. The CPA consists of locally interconnected PE's placed in the nodes of a hexagonal lattice, i.e. each processing cell is connected to 6 neighbours. Such processing grid provides several advantages compared to conventional rectangular one: savings in processing time (fewer neighbours) and hardware implementation (avoiding an overlapping mesh as compared to 8-connected rectangular grid). Due to a parallel structure and asynchronous operation the presented CPA shows high performance in extraction of skeletons of binary images (approximately 45 us for processing a 1024x1024 image).

## 2 SKELETONIZATION

Skeletonization has been demonstrated to be a powerful technique in image processing providing shape representation via extracting important topological features which can be used in object and character recognition and image compression. The above mentioned applications usually require operation at high speeds, so the system performance becomes the critical issue. At the same time, from the system design perspective, it is beneficial to keep the PE structure as simple as possible, so that bigger array size can be achieved within the same silicon area.

### 2.1 Main approaches

There are two main approaches to skeleton extraction, based either on iterative thinning or distance transform. Although the latter methods are less sensitive to a boundary noise, methods based on iterative thinning have several features which become beneficial when the primary goal is a VLSI implementation. In the majority of distance-transform algorithms the data flow between pixels involves transferring a distance value from PE to PE and in some algorithms also transferring a vector value, pointing at a nearest background pixel.

* School of Electrical & Electronic Engineering, The University of Manchester, PO Box 88, Manchester, M60 1QD, United Kingdom;
e-mail: a.lopich@postgrad.machester.ac.uk; p.dudek@manchester.ac.uk

This will require at least $\log_2 N$ (with the image size $N$x$N$) bits for communication between two PE's, whereas in iterative thinning methods this value is invariant to image size. Another disadvantage of distance-based methods is the requirement for bit-vector arithmetic operations with $\log_2 N$ -bit numbers within each cell (as compared with single-bit operations in iterative algorithms). This leads to increased amount of input/outputs and impractical complexity of the processing cell [7]. Altogether the above aspects make the iterative thinning methods better suited for VLSI implementation in terms of logic complexity and area efficiency.

A number of parallel thinning-based algorithms have been developed and explored in previous works [8]. The majority of methods reported in literature use two-distant neighbourhood and template matching to determine if a pixel belongs to a skeleton or not [8]. It leads to a complex overlapping mesh of local interconnections, internal logic complexity and correspondingly area increase. The method presented in this paper overcomes this problem by transferring data only between the nearest neighbours. Each PE shares only two bits of data with its neighbours, which provides sufficient information to achieve skeletons with preserved object topology and connectivity. At the same time the iterative process can be replaced by an asynchronous wave-propagation operation.

### 2.2 Solution

The proposed method has been inspired by the Hildich's algorithm [9]. The input binary image consists of background pixels (logic '1') and object pixels (logic '0') on a hexagonal lattice. The neighbourhood for every PE is defined as follows: $N(P_{ij}) = \bigcup_{k=0}^{5} \{P_{ij}^k\}$. In other words $N(P_{ij})$ is represented by six pixels nearest to $P_{ij}$ as shown in Fig.1(a). Let us define three functions:

$$1.\ B_{ij} = \sum_{k=0}^{5} \bar{P}_{ij}^k \qquad (2.1)$$

$$2.\ A_{ij} = \left| sign\left(\sum_{k=0}^{5} (P_{ij}^k \wedge \bar{P}_{ij}^{(k+1) \bmod 6}) - 1\right) \right| \qquad (2.2)$$

$$3.\ C_{ij} = \prod_{k=0}^{5} (P_{ij}^{(5+k) \bmod 6} \vee P_{ij}^k \vee P_{ij}^{(k+1) \bmod 6} \vee A_{ij}^k) \qquad (2.3)$$

An object pixel is then considered to be not a skeletal point and is removed if all three following conditions are satisfied:

1) $1 < B_{ij} < 6$; (not isolated, internal or endpoint);
2) $A_{ij} = 0$; (not a junction);
3) $C_{ij} = 1$; (prevention of two-pixel lines vanishing).

Each PE has access to the binary values $P_{ij}$ and $A_{ij}$ of all its neighbours and evaluates expressions (2.1)-(2.3) when triggered by a removed neighbour.

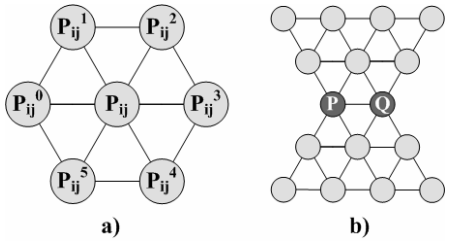


Figure 1: a) Local hexagonal PE neighbourhood $N(P_{ij})$; b) A two-pixel wide line vanishing problem.

Initially all the background pixels are activated, and thus the propagation starts from object's border and spreads towards its interior. The information about the two-distant neighbourhood is implicitly derived from the third condition, by which we prevent situation, when two opposite pixels in a two-pixel wide line simultaneously vanish, because of the mutual consideration of opposite pixel to be internal (Fig.1(b)). The algorithm will produce 2-pixel wide skeletons. Single-pixel wide skeletons can be achieved afterwards with a single synchronous iteration.

## 3. DESIGN IMPLEMENTATION

When designing a VLSI implementation of the array trade-offs between speed, area and circuit complexity need to be considered. The problem with cumbersome interconnections has been eliminated by employing a hexagonal lattice as a processing grid and implementation of an algorithm that requires only local neighbourhood information. The usage of a local neighbourhood doesn't require additional memory elements and comparison logic. Thus an additional combinatorial logic for calculation of $C_{ij}$ value offers a reasonable compromise in terms of performance and area. All the PE's in the CPA operate asynchronously, independently from each other. There is only one synchronization signal, used for uploading an image and resetting internal logic for every PE.

### 3.1. Processing Cell architecture

The structure of the processing cell is shown in Fig. 2. Every PE consists of 14 latches for input/output values, a combinatorial block (CB) for calculation of the values $B_{ij}$, $A_{ij}$, $C_{ij}$ (Eq. 2.1-2.3) and a control unit (CU), responsible for PE self-timing (clocking internal logic, generating an activation pulse). The CU (Fig.3) is a four-state asynchronous Mealy state-machine. Pulses, generated by the CU, update input latches and then, after passing through buffer (delay) elements, update output latches and trigger the CU, shifting it to an appropriate next state.

Each PE shares only two single-bit values with its neighbours: pixel value $P_{ij}$ and parameter $A_{ij}$. The PE input signals consist of $P_{ij}$ and $A_{ij}$ values and activation signals from 6 neighbours as well as a 'Start' signal (global) and pixel input (individual for each PE). The PE is activated when it receives an activation signal from any of its neighbours.

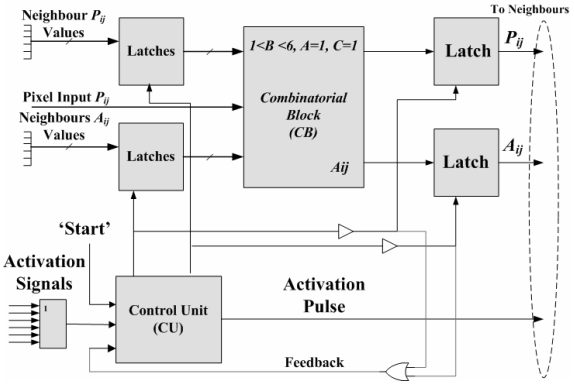


Figure 2: Block diagram of PE

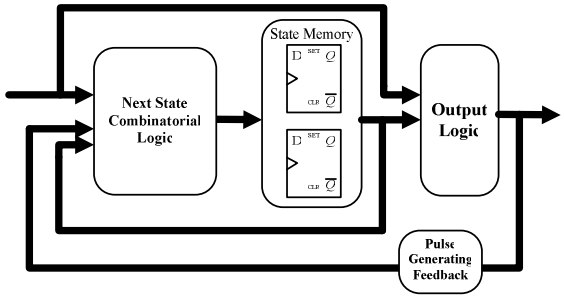


Figure 3: Control Unit block diagram.

In this case the CU generates two sequential clock pulses. The first pulse updates latches with neighbours' $\boldsymbol{P_{ij}}$ values and then the latch containing the value $\boldsymbol{A_{ij}}$ of the current PE. The second pulse updates latches with neighbours' $\boldsymbol{A_{ij}}$ values and then the latch with current pixel value $\boldsymbol{P_{ij}}$. If at any stage of processing the pixel value $\boldsymbol{P_{ij}}$ has been changed to logic '1' (i.e. the pixel doesn't belong to a skeleton and has been removed), then this PE forms an activation pulse, sends it to all its neig hbours and the CU is set into such a state that the pixel will not respond to any input changes, unless the 'Start' signal will be set into '1', indicating a new frame. The 'Start' signal resets the CU and loads a new pixel value. If the pixel belongs to a background (the value is '1') then on the negative edge of the 'Start' signal the PE will generate an activation pulse. Thus all the background pixels are initially activated. In this way the activity spreads in an asynchronous trigger-wave manner across the array and processors perform operations only when required. The speed of the propagation is determined only by the PE's processing time.

### 3.2. Self-timing issues.

Due to the combinatorial path delay in the CB, it was necessary to implement additional logic to handle this delay. Two main methodologies for this case provide either delay-insensitive or bounded-delay solution. The delay-insensitive scheme implies that the correct logic operation is achieved by means of transition signalling and does not depend on propagation delays, i.e. it is speed-independent. However all the delay-insensitive techniques require either a handshaking protocol implementation or double-rail signal coding. Such solutions significantly increase the design size, which also imposes additional delays. This, in our case, makes asynchronous operations inefficient due to its logic complexity and size. Taking these facts into account we have chosen a bounded-delay approach to be implemented for managing the CB delays. Two fixed-delay buffer elements were created for pulse signals, thus ensuring that output registers latch the updated data from the CB. The benefit of this approach over delay-insensitive method is that its VLSI implementation (especially in custom silicon) is relatively simple. For example the solution used in [10] consists of six transistors only and offers a voltage controlled buffer element. Due to the fact that the skeleton extraction is accomplished in an asynchronous manner the "quality" of the resulting skeleton is conditioned by the uniformity of the activity propagation velocity. That is, the less dispersion in processing speed of PE's, the closer (in terms of Hausdorff distance) the asynchronous result will be to its synchronous analogue. Asynchronous extraction can also introduce topological errors. If the maximum difference in processing time between two PE's activated simultaneously is $\Delta t_{max}=t_{max}-t_{min}$, then in order to avoid additional skeletal artefacts the object must not contain the ball of radius $R=[t_{min}/\Delta t_{max}]$ (which we call a critical size) or bigger.

## 4. FPGA REALISATION.

The prototype of the chip has been developed and evaluated on a Xilinx Spartan3 xc3s1500 FPGA in order to provide a behavioural proof of the design and to estimate the processing characteristics. Each PE has been routed and mapped manually and the whole array has been built from PE blocks. In this way we have achieved well controlled timing characteristics for every PE and the same propagation delay between PE's, which has provided uniformity in wave propagation speed and correspondingly skeleton quality. The timing characteristics for different size arrays are presented in Table 1.

| **Array Dimension** | Max. Processing Time Per Frame, ns | Avg. Layer Eroding Time, ns | Max. Processing. Time Difference, ns | Max. Frame Rate, frame/s | Critical Size, pixels |
|---|---|---|---|---|---|
| 16x16 | 584 | 95 | 4 | 1,712,328 | 23 |
| 32x32 | 1349 | | | 741,289 | |
| 64x64 | 2821 | | | 354,484 | |
| 86x86 | 3846 | | | 260,010 | |
| 128x128 | 5805 | | | 172,265 | |

Table 1: Timing results for different size arrays.

The designed 20x20 array has occupied 12387 slices of the target FPGA, which comprised 89% of available slices. It has also consumed 31% of available flip-flops and 71% of LUTs. Each PE has correspondingly occupied 30 slices, 20 flip-flops and 50 LUTs. The CB in FPGA realisation has occupied 14 slices using 24 logic cells. The maximum combinatorial delay path

was 11.9ns. Considering that the maximum delay of input latches was 0.203 ns, the buffer element was designed to provide a pulse skew of not less than 14ns.

It can be seen that the processing time linearly depends on array dimension, i.e. $t_p=O(N)$ for $N$x$N$ array. Therefore an approximate speed for bigger arrays can be easily derived. All the presented data is obtained with the worst-case delay conditions from post-place and route simulations.

## 5. RESULTS AND DISCUSSION

A set of different images has been processed on the designed CPA. The result skeletons, shown in Fig. 4, preserve topology of the objects and convex angles on polygonal shapes (see Fig.4 (d, e, f)).

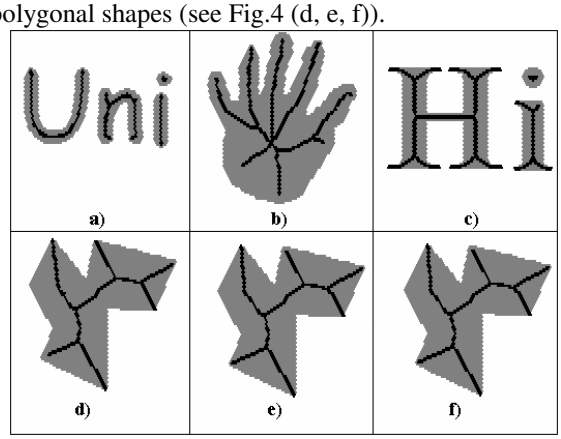


Figure 4: Skeletons and original images.

Possible shift of skeletal legs due to random delay variations comprised only one pixel (Fig.4 (d, e, f)). In other words if $S$ is an intersection of a number of skeletons $S_i(I)$ of the same image $I$:

$$S = \bigcap_{i=0}^{\infty} S_i(I)$$

then $S$ will also form a skeleton of the image $I$ with preserved connectivity. It should be noted that the quality of skeletons is correlated with timing parameters of PE's and object size and shape. For example if the object exceeds the critical size (which depends on PE's timing characteristics) some undesired artefacts may appear.

Although iterative methods, including the method presented here, provide only approximate skeleton position compared to the theoretical definition, it is necessary to consider the fact that the majority of the image processing applications, that involve skeleton extraction, do not require absolute accuracy of the skeletons. For example, high-level methods for object or letter recognition (neural networks, original-sample distance analysis, etc.) allow certain dispersion in the original data due to noise influence, different physical position of objects, etc. At the same time skeletonization is usually one of the pre-processing routines, so the speed of its execution becomes significant. As we can see from Table 1 asynchronous implementation allows skeleton extraction to be accomplished at ultra-high speeds.

## 6. CONCLUSIONS.

The application specific CPA that allows real-time asynchronous extractions of skeletons of binary images has been presented. The designed array attempts to combine processing accuracy, high speed, simple PE structure and respectively small area. The CPA consists of interconnected processing cells arranged in a hexagonal manner. An FPGA implementation of the array (aimed to proof the concept and to estimate behaviour and timing properties) has been realized on Xilinx xc3s1500 FPGA. The timing characteristics of evaluated arrays and results have been presented. The skeletons obtained by parallel asynchronous algorithm have minimum dispersion within predicted range and preserve all the object topological properties. The execution speed is significantly higher than that of contemporary vision systems. The architecture is scalable and the processing time is linearly dependant on the array dimension. In general it has been demonstrated that an asynchronous implementation of wave-propagating operations provides a significant performance increase in image processing applications.

## References


[1] Dudek, P. *A 39x48 general-purpose focal-plane processor array integrated circuit*. in *2004 IEEE International Symposium on Circuits and Systems - Proc., May 23-26 2004*. 2004.

[2] Takashi Komuro, et al. *A Digital Vision Chip Specialized for High-speed Target Tracking.* IEEE Trans. on Electron Devices, 2003.**50**(1):p.191-199.

[3] A. Gentile, et al. *Real-Time Image Processing on a Focal Plane SIMD Array*. in *Proc. of the 7th Int. Workshop on Parallel and Distributed Real-Time Systems*. April 1999. San Juan, Puerto Rico.

[4] Dudek, P. *Fast and Efficient Implementation of Trigger-Wave Propagation on VLSI Cellular Processor Arrays*. in *CNNA2004*. 2004. Budapest, pp. 117-122.

[5] 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 Tr. VLSI Systems, 1996.**4**(3):p. 322-335.

[6] Lopich, A., Dudek, P. *Asynchronous Cellular Processor Array For Skeletonization Of Binary Images.* in *Proc. PREP2005*. 2005. pp. 32-33

[7] Sudha, N., *Design of a cellular architecture for fast computation of the skeleton.* Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, 2003. **35**(1): p. 61-73.

[8] Lam, L., et al. C.,Y.,, *Thinning methodologies-a comprehensive survey.* IEEE Trans. Pattern Analysis and Machine Intelligence, 1992. **14**(9): p. 869-885.

[9] Hilditch,C.,J., *Linear skeletons from square cupboards.* Machine Intelligence, 1969.**4**:p.403-420.

[10]Dudek, P., S. Szczepanski, and J.V. Hatfield, *A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line.* IEEE Journal of Solid-State Circuits, 2000. **35**(2): p. 240-7.