An Asynchronous Cellular Logic Network for Trigger-Wave Image Processing on Fine-Grain Massively Parallel Arrays

Piotr Dudek, *Senior Member, IEEE*

***Abstract*—Massively parallel processor-per-pixel single-instruction multiple data arrays are being successfully used for early vision applications in smart sensor systems; however, they are inherently inefficient when executing algorithms involving propagation of binary signals, such as the geodesic reconstruction. Yet, these algorithms, at the interface between pixel-level and object-level image processing, should be implemented on the vision chip to facilitate data reduction at the sensor level. A cellular asynchronous network is presented in this paper, which can be used to execute binary propagation operations. The proposed circuit is optimized in terms of speed and power consumption. In 0.35-μm technology, the simulated propagation speed is 0.18 ns per pixel and the total energy expended per propagation is 0.37 pJ per cell. In this brief, implementation issues are discussed and simulation results including image processing examples are presented.**

***Index Terms*—Cellular arrays, image processing, parallel architectures, VLSI.**

I. Introduction

LOW-LEVEL image processing problems exhibit a high degree of parallelism and can be efficiently executed on massively parallel processor arrays, operating according to the single-instruction multiple data (SIMD) principle. In particular, digital [1], [2] and analog [3], [4] processor arrays have been demonstrated which not only exploit the fine-grain parallelism of the algorithms, assigning a processing element to each single pixel in the image, but also integrate image sensors for parallel optical image input. Ideally, such "smart sensors" should perform some medium-level image processing algorithms as well, so that the amount of data transmitted off-chip is significantly reduced [5], [6]. This involves the transition from pixel-wise local operations to global (e.g., object-based) operations.

A computer vision algorithm often requires a selection of one of the objects in the image, identified by a marker, for further processing. In the case of binary images, this operation is called "geodesic reconstruction" and is extensively used as a part of many image processing routines. Geodesic reconstruction, illustrated in Fig. 1, can be easily implemented on an SIMD array via an iterative application of the following Boolean equation for each pixel:

$$y = (y_{\mathrm{N}} + y_{\mathrm{E}} + y_{\mathrm{W}} + y_{\mathrm{S}}) \cdot u \tag{1}$$

where u are elements of an array $\mathbf{u}$ corresponding to pixels in the input image (objects are represented by logic "1," background by logic "0") and where y are the pixel elements of an array $\mathbf{y}$ corresponding to the "next state" of the system, while y_{N}, y_{E}, y_{W}, and y_{S} are the pixel's neighbor elements of the array $\mathbf{y}$ (in the North, East, West, and South direction, respectively) corresponding to the "present state" of the system. An appropriate boundary condition $y =$"0," is also required for neighbors outside the array. The state array $\mathbf{y}$ is initially set to "0" everywhere, except for the pixel at the marker location where it is set to "1." The iteration of (1) produces a stable solution after a number of iterations. In this simple case, the von Neumann connectivity (4-neighbors) in the pixel-wise SIMD array is assumed, and the operation could be formulated in terms of cellular automata operation as follows: "set next state to 1 if the state of at least one of the immediate neighbors is equal to 0 and the input is equal to 1." As an example, consider a series of illustrations in Fig. 1, depicting the evolving state image $\mathbf{y}$ obtained as a result of the iteration of (1), with input $\mathbf{u}$ shown in Fig. 1(a). The algorithm works via a propagation of a binary wave, which starts at the marker location, and spreads to fill the entire object. Apart from object reconstruction, this "flood-fill" operation is commonly used to detect holes in binary objects, as illustrated in Fig. 2 (in this case, the "object" being reconstructed is the background of the original image and the operation is initiated along the boundary).

Although the propagation algorithm is simple, yet, the implementation of this algorithm on a synchronous SIMD machine is not efficient. Firstly, it takes a long time to complete. The number of required iterations is usually large and it depends on the shape and the size of the object in the input image. Without prior knowledge about the input image, either an arbitrary maximum number of iterations (possibly larger than the dimension of the array) is applied, or a test (involving global logic operation) is performed at arbitrary intervals to detect whether the propagation has stopped. Secondly, since the operation described by (1) is repeatedly executed by all the processors in the SIMD array, the propagation algorithm is not efficient in terms of power consumption. Consider that each pixel in the image $\mathbf{y}$ needs to change only once during the entire propagation, i.e., in an optimized case, only the logic functions corresponding to

Manuscript received July 6, 2004; revised September 23, 2005. This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC), U.K. This paper was recommended by Associate Editor W. Porod.

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

Digital Object Identifier 10.1109/TCSII.2006.869916

1057-7130/$20.00 © 2006 IEEE

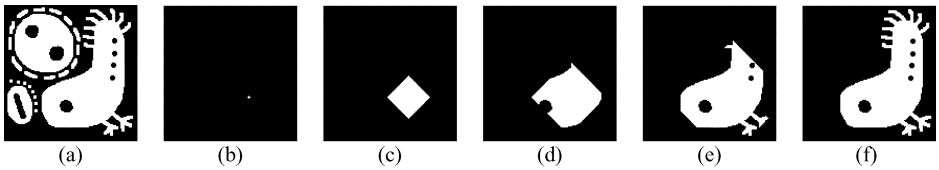

Fig. 1. Geodesic reconstruction of binary objects via the propagation method. (a) Input image. (b) Initial state—location of the marker. (c)–(e) Intermediate results. (f) Reconstructed object; logic "1" (objects) is represented by white pixels, logic "0" (background) by black pixels.

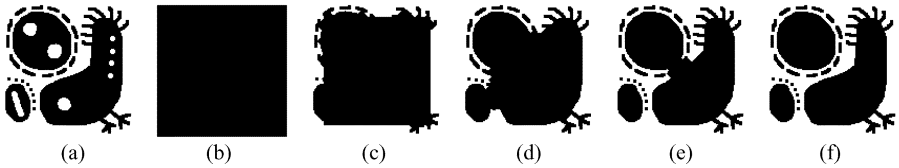

Fig. 2. Hole filling via the propagation method. (a) Input image. (b) Initial state. (c)–(e) Intermediate results. (f) Processing result.

pixels at the front of the wave propagating in $\mathbf{y}$ need to be evaluated at each iteration step.

Yet, since binary propagation is often used in real-time computer vision, it is important that it can be executed quickly and efficiently on general-purpose image processing arrays. The inherent inefficiency of SIMD implementation of propagation operations was discussed in [7], where an asynchronous (continuous-time) solution to the problem was presented. A global logic unit, capable of performing asynchronous propagation operations, was presented in [6]. Another continuous-time approach, based on the propagation of a "trigger-wave" in an excitable medium [8] was demonstrated using analog cellular neural networks (CNNs) [9].

In this brief, an asynchronous Boolean network is presented [10]. This network is intended to complement a pixel-parallel SIMD array processor, and provide it with a facility to rapidly and efficiently execute geodesic reconstruction and other trigger-wave propagation based operations (e.g., hole filling). In contrast to previously reported solutions, the circuit is optimized in terms of its power consumption and the speed of operation. The circuit is presented in Section II, the efficiency of the implementation is discussed in Section III together with some circuit alternatives, circuit simulations, for example, image processing operations and a discussion of these results are presented in Section IV, while Section V concludes the paper.

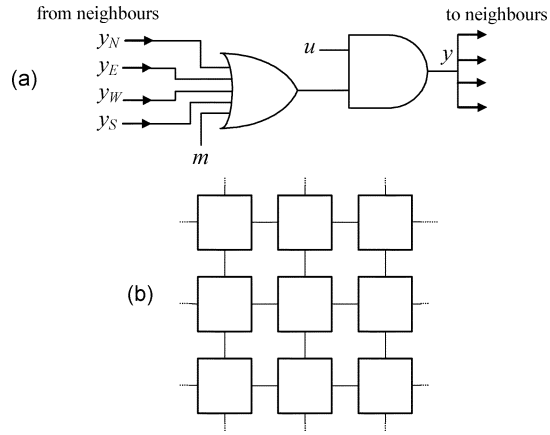

Fig. 3. (a) Logic diagram of a single cell in the ACLA. Output $\mathbf{y}$ of each cell connects to the inputs of its four nearest neighbors. (b) Array topology.

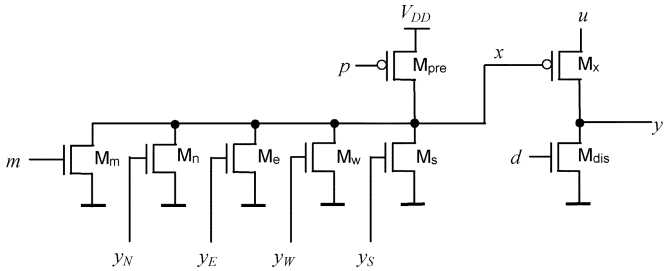

Fig. 4. Schematic diagram of a single cell in the ACLA.

II. Asynchronous Cellular Logic Array

A conceptual logic diagram of the proposed asynchronous cellular logic array (ACLA) is shown in Fig. 3. The logic function implemented by each cell circuit is that of (1); however, instead of a synchronous iteration there is a direct coupling between the output y of one cell and the inputs of its neighboring cells, in the two-dimensional (2-D) network. This coupling facilitates the asynchronous propagation. It should be noted, that the functionality of the ACLA could be achieved by configuring the circuit presented in [6]; the circuit discussed in [7] also achieves equivalent operation. Here, a compact, high-speed and low-power circuit implementation of this operation is proposed. The cell is implemented using the dynamic logic circuit shown in Fig. 4, and a domino-effect is exploited to optimize the power efficiency of the entire network. The operation of the circuit is as follows.

A. Phase I—Precharge

Initially the inputs u and m are set to logic "0" and the node y is discharged to ground (i.e., logic "0") by closing the switch $\mathrm{M_{dis}}$. Then, the node x is charged up to V_{DD} (i.e., logic "1") by closing the switch $\mathrm{M_{pre}}$. The switches $\mathrm{M_{dis}}$, and $\mathrm{M_{pre}}$ are then opened. At this stage, we have $x =$"1" and $y =$"0" and these nodes are put in a high-impedance state. The corresponding pixel of the input image is applied to the input u (the "object" pixels, where the wave will propagate are represented by logic "1"), but since $x =$ "1" switch M_x is open, and so the logic values at y and x remain as they were. At this stage, the circuit is prepared for the asynchronous propagation.

B. Phase II—Propagation

The propagation is initiated by applying logic "1" to the input m of one of the cells, in the array, thus indicating the location of the marker. (Note that a "marker image" could also be used by applying $m =$"1" to several cells). Applying $m =$"1" will result in triggering the propagation, since x becomes "0" at the marker location (the node x is discharged via M_m), which in turn leads to the output y switching to "1" (node y charged up via M_x) if the input u at this location is equal to "1." The output y going high will then trigger neighbor cells, their x nodes being discharged to "0" via transistors $\mathrm{M_N}$, $\mathrm{M_E}$, $\mathrm{M_W}$, and $\mathrm{M_S}$. This will in turn lead to their y nodes being charged up to "1," which will trigger further neighbor cells and so forth. Note, that this propagation is conditioned by the state of the input u at each cell, since the node y can only be charged up to "1" if the input u is equal to "1."[1] Otherwise (when $u =$"0") the cell output y remains equal to "0." The propagation continues in this fashion until the trigger-wave reaches all cells belonging to objects defined by $\mathbf{u}$, marked by $\mathbf{m}$. The result of the processing can then be then read from the output $\mathbf{y}$.

III. Implementation Issues

While the operation of the proposed ACLA can be seen as a straightforward asynchronous implementation of (1), the essential feature of the circuit in Fig. 4 is that it minimizes the overall power consumption associated with trigger-wave propagation. There are only two nodes per cell (x and y) where logic functions are evaluated, and each of the circuit nodes is charged up to V_{DD} and discharged to ground only once (at most) during the entire propagation. There are no static currents (apart from leakage currents) in the circuit, and at no point (even during switching) is there a direct path from V_{DD} to ground. The maximum total energy per cell required to carry out the operation can therefore be estimated as equal to

$$E = \sum_i C_i V_{\mathrm{DD}}^2 \tag{2}$$

where the capacitances C_i are the capacitances of the six circuit nodes (x, y, d, p, u, and m) to ground. In most cases, the total energy will be smaller, as it depends on the number of 1's and 0's in the input, output, and marker images in each particular case.

The node capacitances are mostly due to transistor gate and drain capacitances (to minimize these minimum allowable size transistors should be used), although in practical implementations the effect of parasitic capacitances of wires etc. has to be accounted for. This will not, however, change the fact that the proposed method of trigger-wave generation is, in terms of its power efficiency, very close to the limit of what can be achieved in the CMOS technology.

The network could be further optimized by using complementary cell circuits shown in Fig. 5, distributed across the array according to a chequerboard pattern. Each "n-type" cell connects to four "p-type" neighbors and vice-versa. The propagation follows the domino-effect of charging-down the output nodes of "n-type" cells, and charging-up the output nodes of "p-type" cells. The inputs u and markers m should also be inverted for half of the pixels in the array, according to the chequerboard pattern. In this configuration, the maximum number of nodes that is charged and discharged in one propagation operation is reduced to the absolute minimum (with one evaluation node per cell only). At the same time, the propagation time is reduced to the absolute minimum (one gate delay per cell only) as well. However, in practice, the time and energy gains obtained during the propagation in such an array would be offset by the need to invert half of the pixels in the input image to produce a chequerboard pattern for stimulation, and similarly, to obtain an output image after the propagation. It is therefore expected that the circuit shown in Fig. 4 will be of greater practical use than the alternatives shown in Fig. 5.

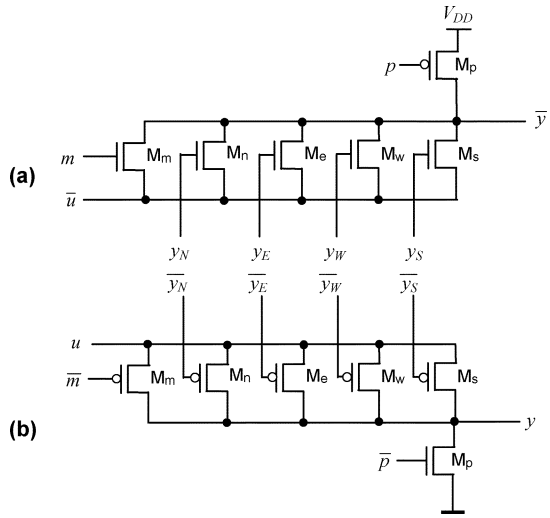

Fig. 5. Two types of cells for optimized ACLA. (a) An "n-type" cell. (b) A "p-type" cell.

It is also worth pointing out, that with a simple modification, shown in Fig. 6(a), the ACLA circuit can be used in a synchronous configuration. In this case switches M_{c1} and M_{c2} are introduced, controlled by global signals clk_1 and clk_2. The introduction of these switches creates dynamic latches. Synchronous operation is achieved by applying the same clock signal to clk_1 and clk_2 during the propagation phase (using two separate global clock lines simplifies the precharge phase, it also allows the circuit to be used both in the synchronous and in the asynchronous mode). While the synchronous configuration reduces the propagation speed, which is determined by the clock, the power consumption of the network remains very low (the only additional energy expended in the synchronous case, in addition to that due to increased node capacitances introduced by the extra transistors, is that required to clock the switches). The synchronous operation might be of use, if a more sophisticated control of the propagation operation is required.

Four separate transistors, controlled by signals c_{N}, c_{E}, c_{W}, and c_{S}, could also be used to control the propagation, as illustrated in Fig. 6(b). The propagation can be then selectively

[1]It should be noted that input u provides current to charge up the output node y, therefore it should be actively driven when in logic state "1." The output stage/buffer required to provide this drive has not been included in the simulations presented in this paper.

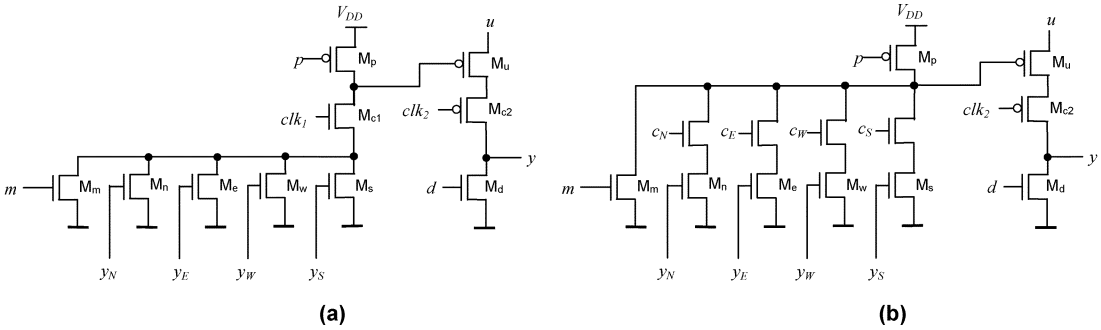

Fig. 6. Modifications to the ACLA circuit. (a) Cell circuit enabling synchronous propagation. (b) Cell circuit with propagation direction control.

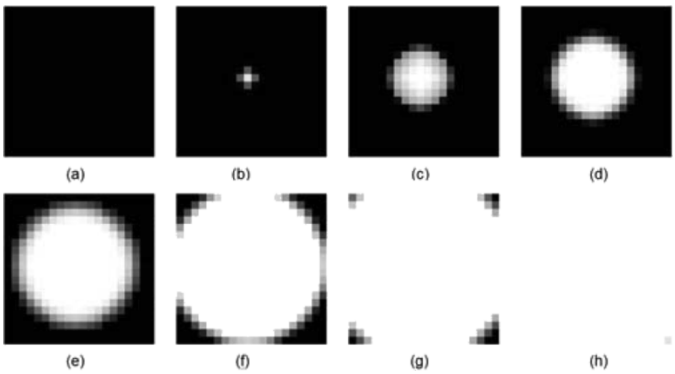

Fig. 7. Propagation in the ACLA. Images (a)–(h) are the snapshots of node voltages y taken at 0.4-ns intervals. Black corresponds to 2.5 V, white corresponds to 0 V.

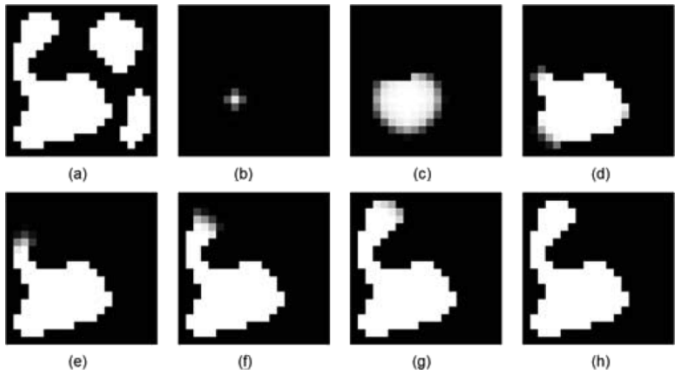

Fig. 8. Geodesic reconstruction. (a) image applied to inputs u. (b)–(h) state of the array (voltages at nodes y) during propagation, taken at 0.5-ns intervals.

disabled/enabled in each direction. This works both for asynchronous and synchronous propagation. In fact, the circuit in Fig. 6(b) offers a compact implementation of a global logic unit proposed in [6].

If the circuit is to operate in a vision chip, i.e., with photosensors integrated on the same substrate, then it is important to ensure that photo-induced leakage currents do not impair the functionality of the network. The asynchronous propagation in the proposed network is very fast, thus the leakage problem should not be very significant. For further protection, and in the synchronous case, appropriate sizes of p-type and n-type diffusion regions (i.e., drains of the transistors in Fig. 4) connected to the evaluation nodes can be used, so that the total effect of photocurrents is that of charging up node x, and discharging node y (see Fig. 3). This will ensure that no false triggering can occur in the network due to leakage currents.

IV. Simulation Results and Discussion

The operation of the proposed ACLA circuit has been confirmed via computer simulations. For that purpose the cell was modeled using minimum-size transistors from a standard 0.35-μm CMOS technology. A 20×20 array of cells has been simulated with SPICE, with a supply voltage $V_{\rm DD}$ set at 2.5 V (although a lower voltage operation is easily achievable, if required).

The effect of the wave propagation in the network is illustrated in Fig. 7. This figure was obtained by plotting the results of the transient SPICE simulation, so that the voltages at nodes y of the cells form images, in which the pixel brightness corresponds to the voltage level, white is 2.5 V (logic "1"), black is 0 V (logic "0"). In this simulation, the inputs u were set to "1" and the propagation was triggered by setting a single-pixel marker in the centre of the array. The state of the array, at various points in time can be observed in Fig. 7. It is interesting to note, that the boundary of the propagating wavefront forms a circular shape, rather than a "diamond" shape, which is obtained in a synchronous system implementing iteratively the (1), as could be seen in Fig. 1. This can be simply explained by the fact, that the speed of the OR gate in the cell circuit is increased if a greater number of its inputs is in logic state "1," since more than one transistor is then discharging the node x. Therefore, the cells at 45° from the centre of the marker are switching somewhat faster than those positioned at right angles. A more detailed insight is of course obtained by considering the magnitudes of the transistor currents and node voltages during the transition period, i.e., analog circuit analysis, as shown by the SPICE simulation. It should be also noted, that in a physical implementation, the transistor mismatch, noise, and asymmetry in layout will lead to variation in propagation speed, affecting the shape of the wavefronts during propagation. However, this will not affect the final (stable) result of the propagation operation.

The geodesic reconstruction operation performed by the proposed circuit is illustrated in Fig. 8. The input image containing several objects was applied to the network, while the propagation was triggered by placing a marker inside one of the objects. The result of the geodesic reconstruction operation is clearly demonstrated.

Closed curve detection (or hole filling)—is shown in Fig. 9. In this case the marker was placed in the bottom-right corner of the image (although in practice it would be better to trigger all pixels along the border of the array, to reduce the overall

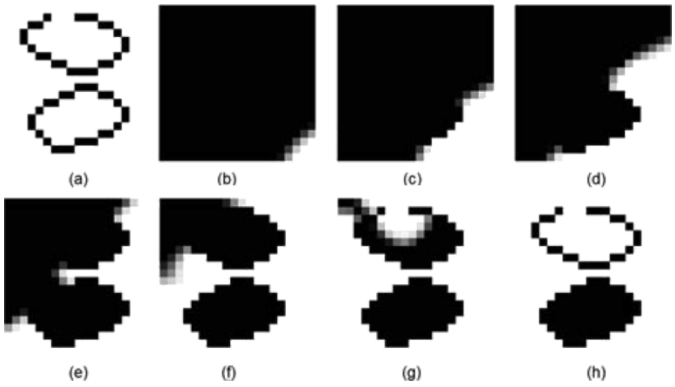

Fig. 9. Closed curve detection. (a) Input image. (b)–(h) state of the array during propagation, taken at 1-ns intervals.

propagation time). The result of the asynchronous propagation operation can be easily combined with the original image, for example to delete the nonclosed curves.

The similarity of the network's behavior to an analog CNN generating a trigger-wave [9] is apparent. Indeed, according to a more general formulation of the CNN paradigm [11], the ACLA can be viewed as a CNN. However, it has to be emphasised, that unlike many CNN implementations [4], [12], which essentially implement logic functions via analog addition and threshold method, the asynchronous cellular logic network proposed in this paper exhibits a critically important feature: that of minimal power consumption. At the same time, the propagation speed is very high. For the SPICE simulations shown in Figs. 6–8, the duration of the asynchronous propagation is between 3 and 7 ns.

The simulation results indicate that the propagation occurs with a speed of approximately 0.18 ns per pixel. The total energy expended per propagation was simulated to be equal to 0.37 pJ per cell. To put these figures in some perspective, consider an array of 128 $\times$ 128 cells, which could be easily integrated as a part of a single-chip SIMD processor array. Using the asynchronous propagation network, the entire trigger-wave operation completes in a few tens of nanoseconds. We shall note, that the obtained speed of operation corresponds to a synchronous SIMD array operating with a 27-GHz clock (assuming, that the operation described by (1) would require 5 clock cycles on a typical bit-serial SIMD processor, with a simple two-argument ALU and one-neighbor-at-a-time connectivity), which would imply an equivalent computational performance of over 440 trillion logic operations per second for a 128 $\times$ 128 array. At the same time, consider a case where the asynchronous array is used in a real-time computer vision application, for example to perform 100 geodesic reconstructions per each image frame, at the processing speed of 30 frames/s. The total power consumption contributed by the asynchronous propagation array to the overall power consumption of the system in this application (3000 reconstructions/s) would be only 1.1 nW per pixel, or 18.2 μW per 128 $\times$ 128 array. These figures do not take into account capacitances that would be created by physical layout, but on the other hand are based on a 0.35-μm digital CMOS technology, which can be easily replaced today by a lower scale technology.

V. Conclusion

An asynchronous cellular logic array has been presented. The circuit employs dynamic logic to facilitate propagation of a trigger-wave, providing means of executing continuous binary morphology operations, such as geodesic reconstruction, with extremely high speed and extremely low-power consumption. The operation of the network has been confirmed by circuit simulations, using a 0.35-μm CMOS technology model. Scaling-down of the technology and power supply voltage will improve the speed and power consumption even further, while reducing the overall area required for the implementation of the circuit. It is expected, that the proposed network will find applications as a "co-processor" to SIMD arrays, particularly those operating as smart image sensors. The proposed cell circuitry is compact enough to be easily integrated into each processing element of a pixel-per-processor array. This integration will dramatically improve the performance, speed, and efficiency of the SIMD array when executing the often-required binary propagation operations.

References

[1] F. Paillet, D. Mercier, and T. M. Bernard, "Making the most of 15 k λ^2 silicon area for a digital retina," in *Proc. SPIE (AFPAEC'98)*, vol. 3410, 1998, pp. 158–167.

[2] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A CMOS vision chip with SIMD processing element array for 1 ms image processing," in *Proc. ISSCC'99*, 1999, TP 12.2.

[3] P. Dudek and P. J. Hicks, "An analog SIMD focal plane processor array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'01)*, vol. IV, May 2001, pp. 490–493.

[4] G. Liñán *et al.*, "Architectural and basic circuit considerations for a flexible 128 $\times$128 mixed-signal SIMD vision chip," *Anal. Integr. Circuits Signal. Process.*, vol. 33, pp. 179–190, 2002.

[5] P. Dudek, "A flexible global readout architecture for an analog SIMD vision chip," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'03)*, vol. III, Bangkok, Thailand, May 2003, pp. 782–785.

[6] J. E. Eklund *et al.*, "VLSI implementation of a focal plane image processor—a realization of the NSIP concept," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 3, pp. 322–335, Sep. 1996.

[7] T. M. Bernard, "Multi-purpose semi-static shift registers for digital programmable retinas," in *Proc. SPIE* , San Jose, CA, Jan. 24–25, 2000.

[8] V. Krinsky, V. Biktashev, and N. Efimov, "Autowaves principles for parallel image processing," *Phys. D*, vol. 49, pp. 247–253, 1991.

[9] C. Rekeczky and L. O. Chua, "Computing with front propagation: active contour and skeleton models in continuous-time CNN," *J. VLSI Signal Process.*, vol. 23, pp. 373–402, 1999.

[10] P. Dudek, "Fast and efficient implementation of trigger-wave propagation on VLSI cellular processor arrays," in *Proc. IEEE Int. Workshop Cellular Neural Networks Their Applications (CNNA'04)*, Budapest, Hungary, Jul. 22–24, 2004, pp. 117–122.

[11] L. O. Chua, "CNN: A vision of complexity," *Int. J. Bifurc. Chaos*, vol. 7, no. 10, pp. 2219–2425, Oct. 1997.

[12] A. Paasio, A. Kananen, and K. Halonen, "A compact computational core for image processing," in *Proc. Eur. Conf. Circuit Theory and Design (ECCTD'01)*, vol. 1, 2001, pp. 337–339.