Paper

FPGA implementation of hardware-oriented reaction-diffusion cellular automata models

Kazuyoshi Ishimura[1 a)], *Katsuro Komuro*[1], *Alexandre Schmid*[2], *Tetsuya Asai*[1], and *Masato Motomura*[1]

[1] *Graduate School of Information Science and Technology, Hokkaido University Kita 14, Nishi 9, Kita-ku, Sapporo Hokkaido, Japan*

[2] *Microelectronic Systems Laboratory, École Polytechnique Fédérale de Lausanne CH-1015, Lausanne, Switzerland*

[a)] *ishimura@lalsie.ist.hokudai.ac.jp*

Received September 1, 2014; Revised November 15, 2014; Published April 1, 2015

Abstract: The implementation of a reaction-diffusion (RD) cellular automata (CA) model on an FPGA is presented. The model generates Turing-like patterns, *e.g.*, striped and spotted patterns observed in marking patterns over animal skins, human fingerprints, etc. Moreover, this model has simple dynamics and generates striped or spotted patterns at its equilibrium that is reached within few cycles, which implies that the model is suitable for hardware implementation. To this aim, a digital processor architecture based on the RDCA model is proposed. Finally, the self-organized patterns generated on the FPGA by the implemented processor are presented.

Key Words: reaction-diffusion, digital processor architecture, FPGA

1. Introduction

Self-organized spotted or striped patterns appear in nature, *e.g.*, the surfaces of animals and fish, fingerprints, in colonies of bacteria and other systems [1–6]. Alan Turing advocated the concept of "diffusion-driven instability" [7] regarding these patterns and their forming processes. This concept indicates that transitions are developed from homogeneous states to spatially inhomogeneous stable states in these systems. The time development of states is described by a sum of reaction and diffusion terms in a system. Reaction represents the production or execution of the state in a local area by summing activators and inhibitors. Diffusion represents a transport process that tends to inhibit any inhomogeneity in the neighboring region. RD equations which are described as partial differential equations exhibit striped or spotted patterns at the equilibrium state.

Various approaches which exploit RD dynamics while reducing nonlinearity to limit computational resources have been investigated. As examples, discretized RD space with cellular automata, discretized parameters and values of dynamics, or both have been presented [8–15]. In this paper, a

RDCA model [16] which has time-dependent diffusion dynamics and discretized reaction dynamics has been used, which has simple dynamics and is suitable for HW implementation. Moreover, we propose a digital processor architecture which generates striped or spotted patterns. Subsequently, we demonstrate that the processor implemented on an FPGA generates self-organized patterns.

This paper is organized as follows. Section 2 describes the RDCA model. Section 3 explains the architecture of the digital RD processor based on the RDCA model. FPGA implementation results are presented in Sect. 4. Section 5 is devoted to the summary and discussion of an application example using the RDCA model.

2. A reaction-diffusion cellular automata model

In the RDCA model, each state of a cell is determined by a sigmoid function and a weighted-sum computation that takes interaction with four adjacent cells into account. The weighted-sum computation implies that activators and inhibitors diffuse in individual diffusion fields, and they are convoluted in each of the cells. Each state in the cells is computed as the difference between the states of activators, u, and inhibitors, v, for each cell (x, y) in the field. For the digital HW implementation, the diffusion dynamics is discretized as

$$\frac{u_{ij}(t+\Delta t) - u_{ij}(t)}{\Delta t} = D_u \frac{u_{i+1j}(t) + u_{i-1j}(t) + u_{ij-1}(t) + u_{ij+1}(t) - 4u_{ij}(t)}{h^2}, \tag{1}$$

$$\frac{v_{ij}(t+\Delta t) - v_{ij}(t)}{\Delta t} = D_v \frac{v_{i+1j}(t) + v_{i-1j}(t) + v_{ij-1}(t) + v_{ij+1}(t) - 4v_{ij}(t)}{h^2}, \tag{2}$$

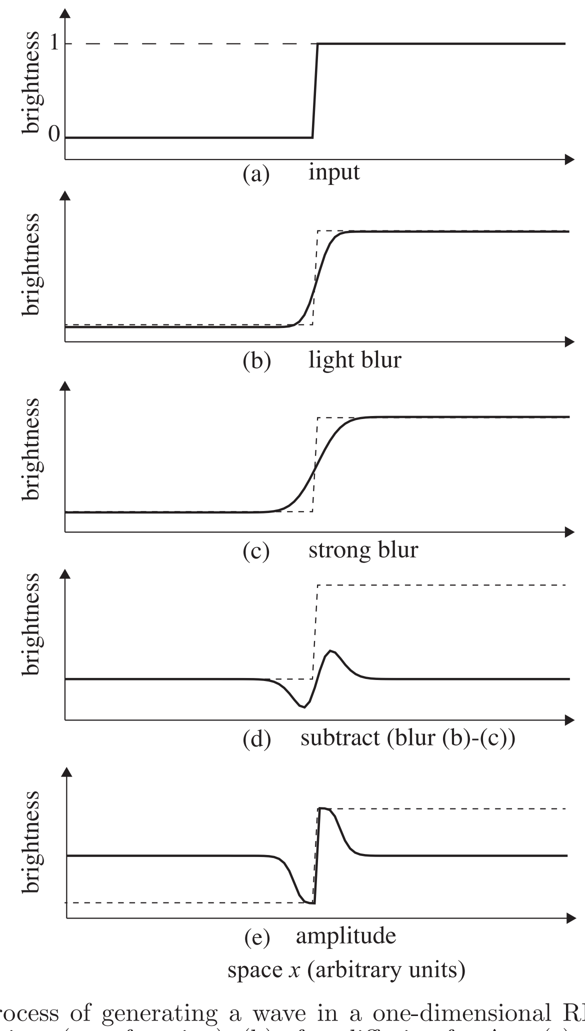


Fig. 1. Process of generating a wave in a one-dimensional RD model: (a) initial conditions (step function), (b) after diffusion for Δt_0, (c) after diffusion for Δt_1 - Δt_0, (d) subtraction of the activator from the inhibitor, and (e) the subtraction in (d) that is amplified by the sigmoid function.

where i and j represent the coordinate of the x axis and the y axis on the square diffusion grid, respectively, h represents the increase of x or y, D_u represents the diffusion coefficient of the activators and D_v represents the diffusion coefficient of the inhibitors. D_v and D_u are selected to satisfy that D_v is larger than D_u. The diffusion equations for u and v are integrated for a time δt. A cell's subsequent state is determined by the value of the sigmoid function of $u - v$. The reaction dynamics is described as

$$\begin{aligned} u(x, y, \delta t(n+1)) &= v(x, y, \delta t(n+1)) \\ &= f(u(x, y, \delta t \cdot n) - v(x, y, \delta t \cdot n) - c), \qquad (3) \\ f(w) &= \frac{1}{1+e^{-\beta w}}, \qquad (4) \end{aligned}$$

where n represents the time step, β represents the measure of steepness of the function, and c represents an offset value.

Figure 1 shows the process of forming a spatiotemporal wave pattern in one-dimensional RD in a diffusion field considering $D = 0.001$, $D_v/D_u = 3.0$, $h = 0.01$, $\Delta t = 0.01$, $\beta = 20$ and $c = 0$. In this paper, the diffusion coefficient D represents the diffusion coefficient D_u of activators. Figure 1(a) shows an initial condition as a step function. Solutions in RD models have a form $D \cdot t$ which indicates that diffusion and time are commutable. Therefore, we can assume $D_v/D_u \equiv \Delta t_0/\Delta t_1$. To impose $D_v/D_u = 3.0$, we set $\Delta t_0 = 15$ and $\Delta t_1 = 45$. After diffusion for Δt_0, the lightly blurred initial step function is temporarily stored, in Fig. 1(b). This stored step function corresponds to the initial step function diffusion with a factor D_u for Δt_0. Then, the stored step function is further diffused for $\Delta t_1 - \Delta t_0$. At time Δt_1, the initial step function has been numerically diffused using coefficient D_u for Δt_1 yielding a strongly blurred step function, which corresponds to the initial step function after a diffusion with D_v for Δt_0 in the mathematical model. As a consequence, the initial step function is blurred for Δt_0 with different diffusion coefficients D_v and D_u in a diffusion field. Figure 1(d) shows the difference of Figs. 1(b) and 1(c), that corresponds to the difference of activators and inhibitors. Finally, this difference is amplified by the sigmoid function in Fig. 1(e). "One update" is defined as the application of the resulting amplified output waveform as input of a new processing.

3. Architectures of a digital RD processor

3.1 One-dimensional RD processor architecture

The architecture of a digital RD processor that is suitable to FPGA implementation of the RDCA model is presented in the following. A one-dimensional RD processor architecture if first presented in Fig. 2, which handles 8 bit value and 120 addresses. As indicated in Figs. 1(b) and 1(c), an input signal is diffused by different diffusion coefficients. Therefore, two blurring filters which have different step functions are implemented in the processor. After double filtering, the blurred input signals and offset value c are subtracted. Finally, the amplified result updates the main memory value. Wave patterns are generated from repeated updates.

The nearest-neighbor kernel shown in Fig. 3(a) is used and expanded to develop the blurring filters. Expanding the kernel range enables emulating a stronger intensity of the diffusion coefficient. The outputs out_0 in kernel 1 and 2 are respectively given by

$$\text{kernel1 out}_0 = (a_{-2} + 4a_{-1} + 6a_0 + 4a_1 + a_2)/16 \qquad (5)$$

$$\begin{aligned} \text{kernel2 out}_0 = (&a_{-4} + 5a_{-3} + 11a_{-2} + 15a_{-1} + 16a_0 \\ &+15a_1 + 11a_2 + 5a_3 + a_4)/80, \qquad (6) \end{aligned}$$

where a represents a cell such that a_0 is centered within the filter kernel. These equations correspond to Eqs. (1) and (2) which explain the diffusion dynamics of activators (u) and inhibitors (v), respectively. Figure 4 shows the architecture of the filter module that uses filter kernels 1 and 2. The values originating from cells which are read from the main memory are shifted right, and stored into registers (Z^{-1}), one by one and obey a clock sequence. The input signal is blurred by filters with kernel 1 or 2 which share some registers in order to save area. Because the blurring filters need different numbers

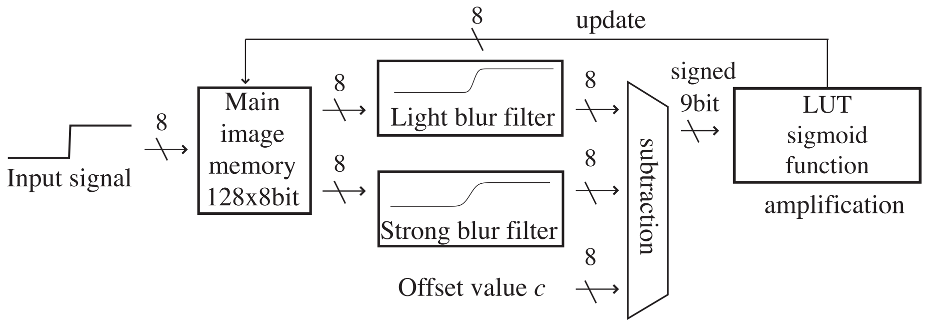


Fig. 2. Architecture of the one-dimensional RD processor.

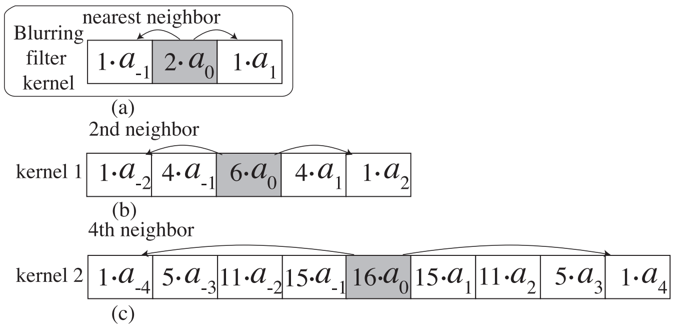

Blurring filter kernel
nearest neighbor
$1 \cdot a_{-1}$ | $2 \cdot a_0$ | $1 \cdot a_1$
(a)
2nd neighbor
kernel 1
$1 \cdot a_{-2}$ | $4 \cdot a_{-1}$ | $6 \cdot a_0$ | $4 \cdot a_1$ | $1 \cdot a_2$
(b)
4th neighbor
kernel 2
$1 \cdot a_{-4}$ | $5 \cdot a_{-3}$ | $11 \cdot a_{-2}$ | $15 \cdot a_{-1}$ | $16 \cdot a_0$ | $15 \cdot a_1$ | $11 \cdot a_2$ | $5 \cdot a_3$ | $1 \cdot a_4$
(c)

Fig. 3. One-dimensional kernel of the blurring filter: (a) nearest neighbor kernel, (b) second-neighbor kernel, and (c) fourth-neighbor kernel.

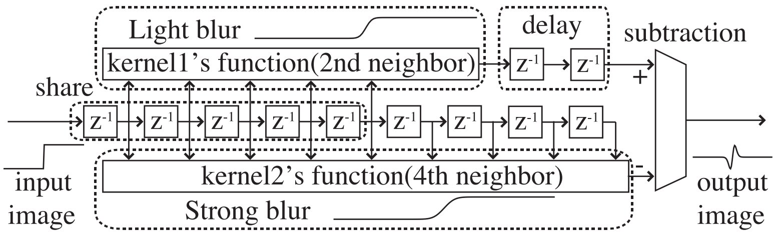
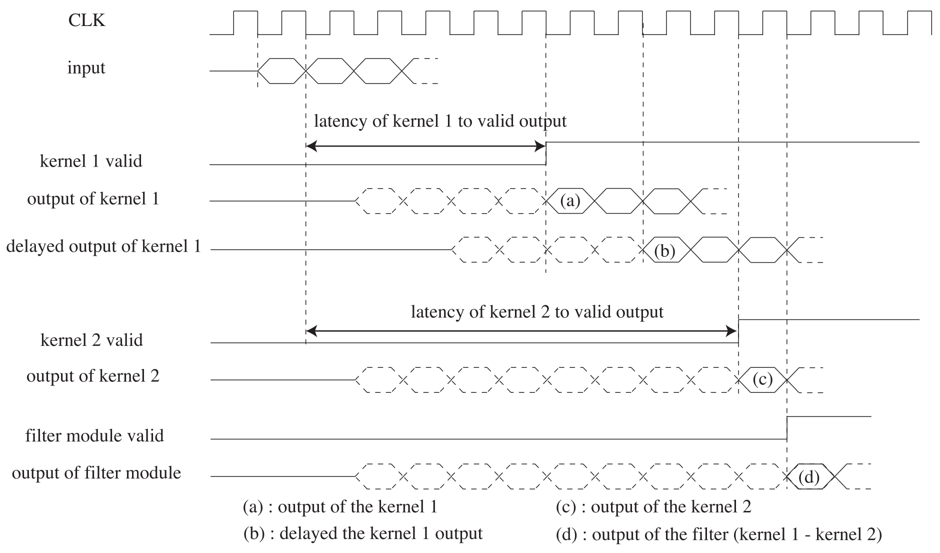


Fig. 4. Data stream architecture of the blurring filter in the one-dimensional RD processor.

of delay cells, the final subtraction process needs timing adjustment registers for kernel 1. Then, these outputs of the filters are subtracted. This subtraction process corresponds to the reaction dynamics (Eq. (3)) without amplification by a sigmoid function and subtraction of an offset variable c which is set to 0. Subsequently, the subtracted signal is amplified by a LUT operation, which functionally performs the sigmoid operation in (Eqs. (3) and (4)). Finally, the amplified signal is stored into the main memory. Hence, the proposed architecture has almost an identical number of controllable parameters as the RDCA model, and only differs from the precision of calculations, *i.e.*, the RDCA model treats floating-point numbers and the FPGA implementation treats fixed-point numbers.

Figure 5 shows the subtraction timings, where the output of kernel 1 is delayed by two clock cycles. Bus *input* represents streaming data which is read from a main memory at each falling edge of clock CLK. The *kernel 1 valid* signal defines the timing of the valid output data from the filter kernel 1. When *kernel 1 valid* signal is low, data streaming out of kernel 1 is invalid. As shown in Figs. 3(b) and 3(c), the center cell of kernels 1 and 2 are shifted in a proportion that corresponds to two clock cycles, in hardware. Therefore, the *delayed output of kernel 1* bus has its timing adjusted for subtraction from the output bus of kernel 2. The *kernel 2 valid* signal defines the timing of the valid output data from the filter kernel 2. The outputs of kernel 1 and 2 are subtracted when the output of kernel 2 is valid. Subsequently, the *filter module valid* signal rises. Finally, the *output of filter module* bus carries data from the blurring filter.



CLK
input
latency of kernel 1 to valid output
kernel 1 valid
output of kernel 1
delayed output of kernel 1
latency of kernel 2 to valid output
kernel 2 valid
output of kernel 2
filter module valid
output of filter module
(a)
(b)
(c)
(d)
(a) : output of the kernel 1
(b) : delayed the kernel 1 output
(c) : output of the kernel 2
(d) : output of the filter (kernel 1 - kernel 2)

Fig. 5. Timing chart of the blurring filters in the one-dimensional RD processor.

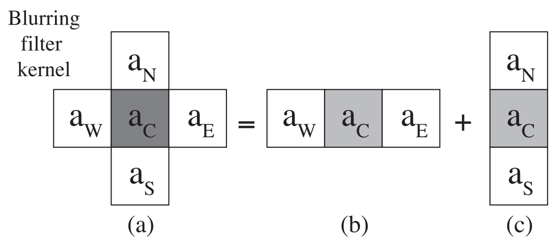


Fig. 6. Kernels of two dimensional blurring filter: (a) nearest neighbor kernel, (b) one-dimensional blurring filter kernel in x direction, and (c) one-dimensional blurring filter kernel in y direction.

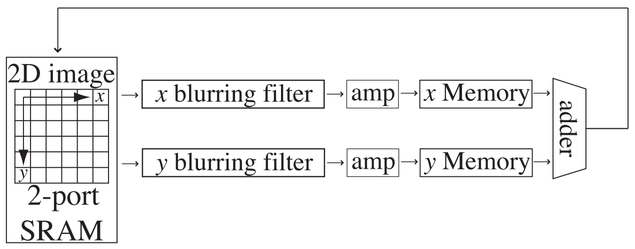


Fig. 7. Architecture of the two-dimensional RD processor.

3.2 Two-dimensional RD processor architecture

The architecture of a two-dimensional RD processor which is an expanded version from the one-dimensional RD processor is presented in the following. Figure 6(a) shows a two-dimensional blurring filter kernel which involves four nearest-neighbor cells. The two-dimensional kernel is implemented in the form of two one-dimensional kernels which handle the blurring process into the x and y directions, respectively, as shown in Figs. 6(b) and 6(c). This technique allows to apply the abovementioned one-dimensional RD processor to the x and y kernels. In order to generate a blurred 2D image, the results from blurring filters into x and y directions are added.

Figure 7 shows the architecture of a two-dimensional RD processor. Using a true dual-port SRAM enables simultaneous reading of two values from different memory addresses. To blur an image data, two one-dimensional blurring filters are required, which are connected to the SRAM outputs and handle the x and y directions. Subsequently, the filtered results are stored into temporary memories to adjust the timing of the 2D blurred image generation process. The sum of values in these temporary memories are stored into the main memory as one update, as shown in Fig. 7. By repeating updates, 2D striped patterns are formed.

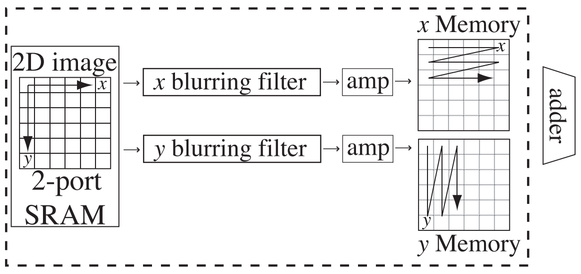


Fig. 8. First stage of the RD process: Read operation, filter processing and writing data into the temporary memories x and y in parallel.

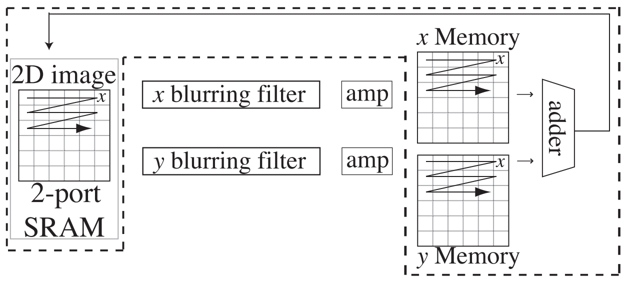


Fig. 9. Second stage of the RD process: Adding x to y, amplifying by LUT, and writing data into the main memory.

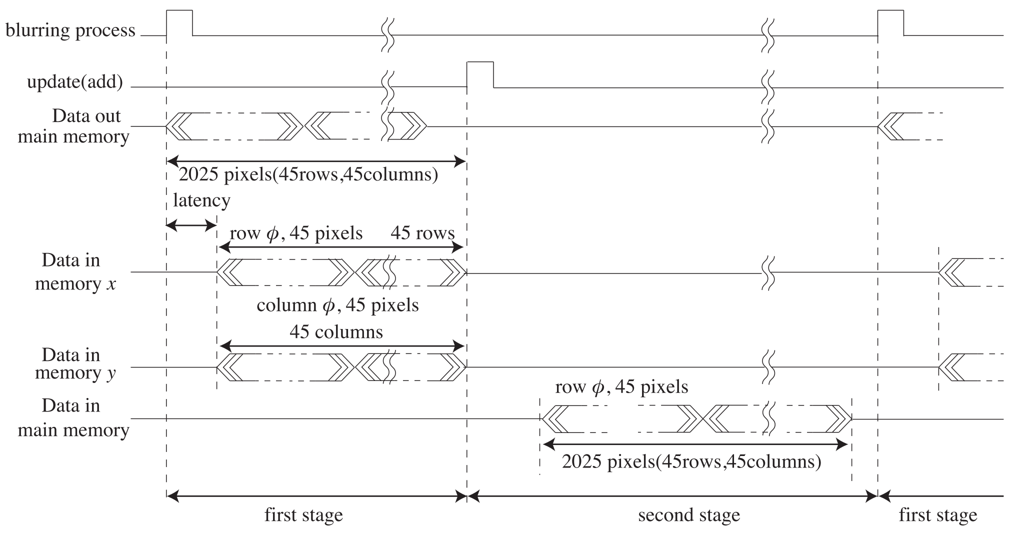


Fig. 10. Timing chart of the 2D RD processor.

The architecture implements two operational steps. First, two values are simultaneously read from the main memory. After blurring and amplifying these values, the filtered results are stored into their temporary memories, respectively, as shown in Fig. 8. In order to read the memory in the x direction, a row address counter is incremented. On the other hand, to read the memory in the y direction, a column address counter is increased by N which is determined as the width of an image, in pixels. When the memory address reaches the bottom of a column, its address changes to the top of the next column. These read operations are repeated until reading the final pixel.

Second, values are read from each temporary memories (x and y) in the x direction by using the same address counter which increments addresses, as shown in Fig. 9. After adding these values, the results are stored into the main memory. A finite-state machine controls the address counters that generate the read and write memory addresses, and the read and write enable signal governing the memories operations.

Figure 10 shows the timing chart pertaining to one update, which consists of a parallel blurring process and summation of blurred data on the 2D RD processor. The *blurring process* signal starts the blur filtering process and the *update* signal triggers the addition of blurred data in x axis and y

axis. The *Data out main memory* signal represents streaming data which consists of a 45×45 pixels image initially stored in the two-port SRAM. The image size is supported by the internal FPGA memory capacity. Fixed boundary operation conditions are set. *Data in memory x* and *Data in memory y* signals identify the blurred data streams that are stored into x and y memories. When the last (2025th) pixel is stored into these memories, the blur filtering completes as shown in Fig. 8. Subsequently, the adding process starts as shown in Fig. 9. The *Data in main memory* is a data stream which is formed by summation of corresponding pixels in memory x and y. One update process is completed when the data is entirely stored into the main memory.

4. Implementation of the RD CA processor

We implemented the RDCA processor on an Altera Cyclone IV platform operating at a frequency of 5 MHz. A one-dimensional RD processor based on the above mentioned architecture has first been implemented. Table I shows the specification of the 1D RD processor. In the main memory, unsigned 8 bit values are stored in each cells. Signed 8 bit values are generated at the output of the filter. The signed 8 bit values are amplified and shifted into positive values (unsigned 8 bit values) in the amplification module. Figure 11 shows temporal evolution results of the RD process, which generates waves from a step function input. The x axis represents the memory address, and the y axis represents the update number. The wave pattern in the first update corresponds to the wave pattern in Fig. 1(e). By repeating updates, wave patterns spread from the center cell.

Subsequently, a two-dimensional RD processor has been implemented. Table I shows the specifica-

Table I. 1D RD processor specs.

Number of cells: 128
Depth: signed 8 bit
Total logic elements: 1,275
Total combinational functions: 1,044
Dedicated logic registers: 593
Total registers: 593
Total memory bits: 1,024

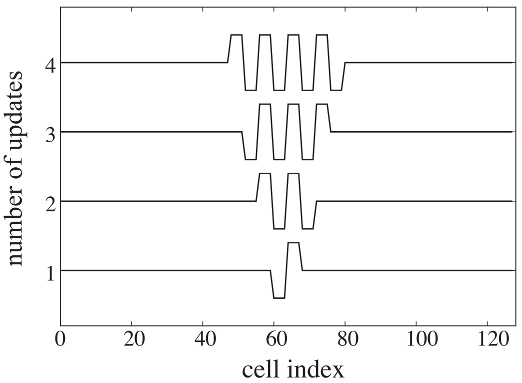


Fig. 11. FPGA implementation results using the one-dimensional RD processor.

Table II. 2D RD processor specs.

Number of cells: 2025
Depth: signed 8 bit
Total logic elements: 2,493
Total combinational functions: 2,205
Dedicated logic registers: 894
Total registers: 894
Total memory bits: 49,152

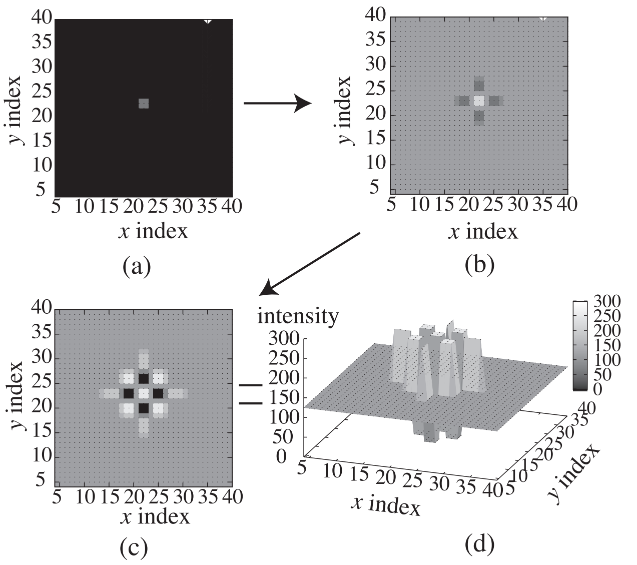


Fig. 12. FPGA implementation results of a two-dimensional RD processor: (a) initial condition (impulse input), (b) first update, (c) second update, and (d) 3D representation of the second update.

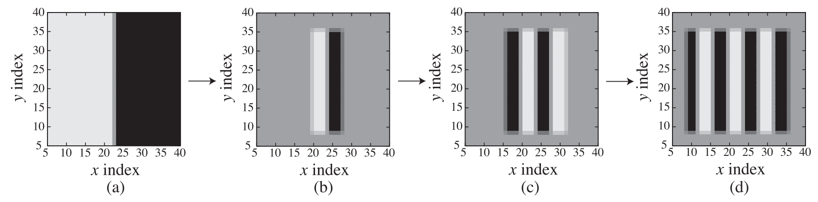


Fig. 13. FPGA implementation results of a two-dimensional RD processor: (a) initial condition (step input), (b) first update, (c) second update, and (d) stable state.

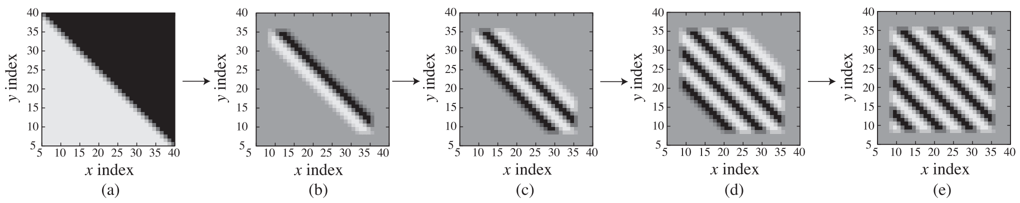


Fig. 14. FPGA implementation results of a two-dimensional RD processor: (a) initial condition (triangular input), (b) first update, (c) second update, (d) third update, and (e) stable state.

tion of the 2D RD processor. To blur an image in parallel, the 2D RD processor requires two filters and amplification units. Therefore, the total amount of logic elements, and combinational functions are approximately doubled with respect to the 1D RD processor resources. In addition to the main memory, two temporary memories are required, that are used to adjust summation timings. Figures 12(a)–(c) show results which are obtained from an impulse input located in the center of the image. Figure 12(d) shows the 3D result in the second update. The pixel values are represented by the intensity (contrasting density) in the z axis direction. First, an impulse is applied to the center as an initial input. Because the amplifier has steep slope to generate RD patterns in a small area, dotted patterns are formed by updating several times. Therefore, from an impulse input, striped patterns will be observed in a RD processor which has extended RD space and a smooth slope amplifier module. Figure 13 shows the results which are obtained from a step function input (Fig. 13(a)). Though the initial input covers all memory addresses, values corresponding to one half of 8 bit are invalid and treated as such after the first update. By repeatedly updating, a wave pattern expands into the right and left directions as shown in Figs. 13(b)–(d). Figure 14 shows the results which are obtained from a triangle step function input. By updating several times, a wave pattern expands to diagonal

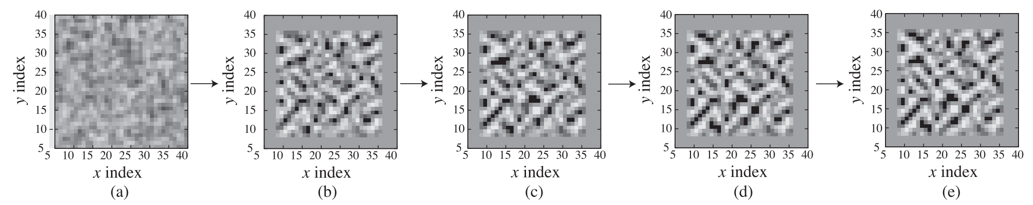


Fig. 15. FPGA implementation results of a two-dimensional RD processor: (a) initial condition (random input), (b) first update, (c) second update, (d) third update, and (e) stable state.

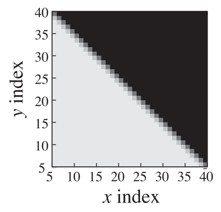
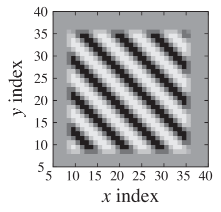
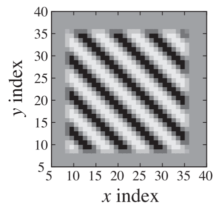
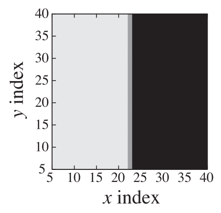
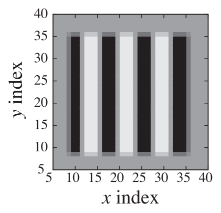
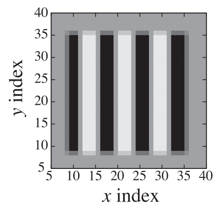
initial input	simulation results	FPGA implementation results	Mean Squared Error (MSE)
			0
			0

Fig. 16. Comparison of RD results. The first column presents the initial inputs; the second column presents the results obtained from the RDCA model computer simulation; the third column presents the results obtained from the proposed model operated on FPGA; the fourth column presents the mean squared error (MSE) calculated between the RDCA model (second column) and the proposed model (third column).

directions as shown in Figs. 14(b)–(d). After the fifth update, the wave pattern stabilizes into a state as shown in Fig. 14(e). Figure 15 shows the results which are obtained from a random input. By updating several times, a random wave pattern is gradually formed as shown in Figs. 15(b)–(d). After the fifth update, a wave pattern stabilizes into a state as shown in Fig. 15(e). The formation of wave patterns by the implemented RDCA processor is confirmed from these experiments.

Figure 16 shows a comparison of RD final results (Figs. 13 and 14), operating the RDCA formal model (second column) and the proposed hardware architecture from two identical initial conditions. The mean squared error (MSE) metrics is adopted to compare the results in a quantified manner and is presented in the fourth column. The MSE is defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \tag{7}$$

where n represents the number of pixels in each image, $\hat{Y}$ represents the estimated image and Y represents the ground truth image. 8bit values within valid 27×27 pixel sub-windows are considered in the calculation (a full-window size is 45×45 pixels). Hence, the maximal MSE value equals 65,025. As observed from null MSE values in the first and second rows, identical RD patterns are generated by the RDCA model and the RD processor.

Figure 17 shows a comparison of RD trials towards reaching stable state, and operating the RDCA formal model as well as the proposed hardware architecture from two identical initial condition patterns consisting of random noise (Fig. 15). The RDCA simulation results (Fig. 17(a)–(e)) and the FPGA measured results (Fig. 17(f)–(j)) are presented as a function of the number of RD updates. After the first update, the MSE is observed at 27,942. By repeating RD updates, the MSE increases.

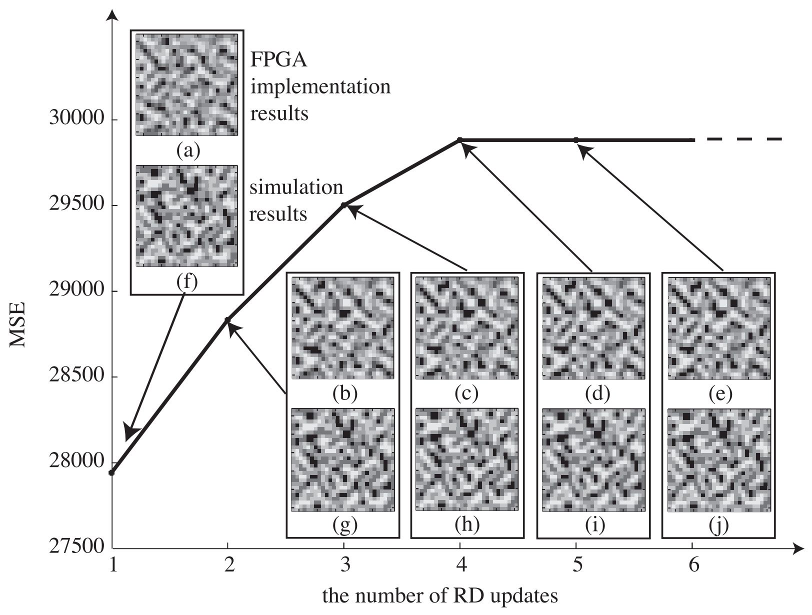


Fig. 17. Comparison of RD results. The horizontal axis corresponds to number of RD updates and the vertical axis corresponds to the MSE computed confronting the RDCA model simulation (a)–(e) and the FPGA implementation results (f)–(j). The line that joins the discrete calculated points is depicted to improve the graphical interpretation and has no physical meaning.

Increasing from the fourth update, the MSE is observed at 29,882 which corresponds to an approximate 46% of the maximal theoretical MSE value of 65,025. This difference arises from a discrepancy in the number representation and calculation between the computer simulation (32bit) with respect to the proposed model implemented on FPGA (8bit). Moreover, errors originate from the floating point representation and calculation used in computer simulations that contrasts to the fixed point calculation used on the FPGA; these errors are further expanded by the sigmoid function. Consequently, significant errors (*e.g.*, $(255-0)^2$ or $(0-255)^2$) in the sigma term in Eq. (7) may occur when striped patterns slightly shift from their high-resolution geometrical expected location. Nevertheless, the nature of the pattern as well as its general and local shapes are preserved.

5. Summary

The hardware architecture of a Reaction-diffusion processor which uses an RDCA model is presented and has been implemented on an FPGA. The RDCA model which is considered in this paper has simple dynamics and reaches an equilibrium of striped pattern within few updates. The model had earlier been implemented in analog CMOS to repair fingerprint images, and has been confirmed suitable for hardware-based applications. A novel steganography method based on Reaction-diffusion had been earlier shown an efficient application using the reaction-diffusion phenomenon [17]. In RD steganography, the RD process is applied to improve data security after hiding a plain text within a random-dot image. The 2D RD processor presented in this paper is suitable to support the realization of RD-based steganography. Furthermore, this hardware can process large images much faster than software processing, thereby opening the way to real-time usage of 2D RD applications.

Acknowledgments

This study was supported by a Grant-in-Aid for Scientific Research on Innovative Areas [2511001503] from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan. The authors gratefully acknowledge the support of Altera University Program.

References

[1] G. Nicolis and I. Prigogine, *Self-organization in Nonequilibrium Systems — From Dissipative Structures to Order through Fluctuations*, John Wiley & Sons, Inc., New York, NY, 1977.

[2] Y. Oono and S. Puri, "Study of phase-separation dynamics by use of cell dynamical systems. I. Modeling," *Phys. Rev. A*, vol. 38, no. 1, pp. 434–453, 1988.

[3] I. Epstein and J. Pojman, *An introduction to nonlinear chemical dynamics*, Oxford University press, Oxford, 1998.

[4] A. De Wit, *Spatial patterns and spatiotemporal dynamics in chemical systems, Adv. Chem. Phys.*, vol. 109, pp. 435–513, 1999.

[5] P. Ball, *The Self-Made Tapestry: Pattern Formation in Nature*, Oxford University press, Oxford, 2001.

[6] J.D. Murray, *Mathematical Biology II (3rd Ed.)*, Chap. 2, p. 75, Springer, New York, 2002.

[7] A.M. Turing, "The chemical basis of morphogenesis," *Phil. Trans. R. Soc. Lond B.*, vol. 237, pp. 37–72, 1952.

[8] D.A. Young, "A local activator-inhibitor model of vertebrate skin patterns," *Math. Biosci.*, vol. 72, pp. 51–58, 1984.

[9] M. Markus and B. Hess, "Isotropic cellular automaton for modeling excitable media," *Nature.*, vol. 347, no. 6288, pp. 56–58, 1984.

[10] M. Gehardt and H. Schuster, "A cellular automaton describing the formation of spatially ordered structures in chemical systems," *Physica D.*, vol. 36, pp. 209–221, 1989.

[11] M. Gehardt, H. Schuster, and J.J. Tyson, "A cellular automaton model of excitable media II. Curvature, dispersion, rotating waves and meandering waves," *Physica D.*, vol. 46, pp. 392–415, 1990.

[12] Q. Ouyang and H.L. Swinney, "Transition from a uniform state to hexagonal and striped Turing Patterns," *Nature*, vol. 353, pp. 610–612, 1991.

[13] J.R. Weimar and J.-P. Boon, "Class of cellular automata for reaction-diffusion systems," *Phys. Rev. E.*, vol. 49, pp. 1749–1752, 1994.

[14] I. Kusch and M. Markus, "Molluse shell pigmentation: cellular automaton simulations and evidence for undecidability," *J. Theor. Biol.*, vol. 178, pp. 333–340, 1996.

[15] Y.-N. Wu, P.-J. Wang, C.-J. Hou, C.-S. Liu, and Z.-G. Zhu, "Turing patterns in a reaction-diffusion system," *Commun. Theor. Phys.*, vol. 45, no. 4, pp. 761–764, 2006.

[16] Y. Suzuki, T. Takayama, I. Motoike, and T. Asai, "Striped and spotted pattern generation on reaction-diffusion cellular automata: Theory and lsi implementation," *Int. J. Unconv. Comput.*, vol. 3, pp. 1–13, 2007.

[17] L. Saunoriene and M. Ragulskis, "A secure steganographic communication algorithm based on self-organizing patterns," *Phys. Rev. E*, vol. 84, issue 5, article no. 056213, pp. 47–60, 2011.