The Data-and-Signals Cellular Automaton and Its Application to Growing Structures

André Stauffer[1]
Moshe Sipper[2]
[1]Logic Systems Laboratory
Swiss Federal Institute of Technology in Lausanne
CH-1015 Lausanne
Switzerland
andre.stauffer@epfl.ch

[2]Department of Computer Science
Ben-Gurion University
Beer-Sheva 84105
Israel
sipper@cs.bgu.ac.il

Abstract In a traditional cellular automaton (CA) a cell is implemented by a rule table defining its state at the next time step, given its present state and those of its neighbors. The cell thus deals only with states. We present a novel CA where the cell handles data and signals. The cell is designed as a digital system comprising a processing unit and a control unit. This allows the realization of various growing structures, including self-replicating loops and biomorphs. We also describe the hardware implementation of these structures within our electronic wall for bio-inspired applications, the BioWall.

Keywords
Cellular automata, digital systems, growing structures, self-replication, biomorph

1 Introduction: Cellular Automata

Cellular automata (CAs) were originally conceived by Ulam and von Neumann in the 1940s to provide a formal framework for investigating the behavior of complex, evolving systems. They are part of the early history of self-replicating machines and of von Neumann's thinking on the matter [8, 13]. Nowadays, CAs still remain the model of choice within the field of artificial self-replication.

Cellular automata are dynamical systems in which space and time are discrete. A CA consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously (in general) in discrete time steps, according to a local interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells. This transition is usually specified in the form of a rule table delineating the cell's next state for each possible neighborhood configuration. The cellular array (grid) is n-dimensional, where $n = 1, 2, 3$ are used in practice [7, 12].

Figures 1a and 2a show the basic two-dimensional automaton cells (ACs) defined for a five-neighbor and a nine-neighbor cellular space, respectively. These cells receive some or all of the states NSI, $NESI$, ESI, $SESI$, SSI, $SWSI$, WSI, and $NWSI$ from the cells to the south, southwest, west, northwest, north, northeast, east, and southeast, respectively. These cells also share their own state SO directly or indirectly with their four or eight neighbors. Such cells consequently deal only with input and output states. They are implemented as sequential machines, resulting from the interconnection of a rule table TAB and a state register REG (Figures 1b and 2b). In order to simplify the design of the cells, the register REG is sometimes functionally sliced into multiple state-variable groups called *fields*. The utilization of such fields also makes the resulting rules of the table TAB much more readable.

This letter introduces a novel cellular automaton, which processes both data and signals. In designing this data-and-signals CA (DSCA) we had two goals in mind: (1)

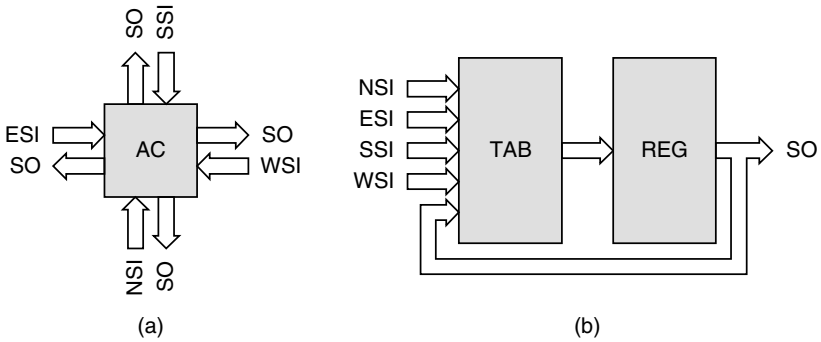


Figure 1. Two-dimensional, five-neighbor CA. (a) Basic cell. (b) Rule table TAB and state register REG.

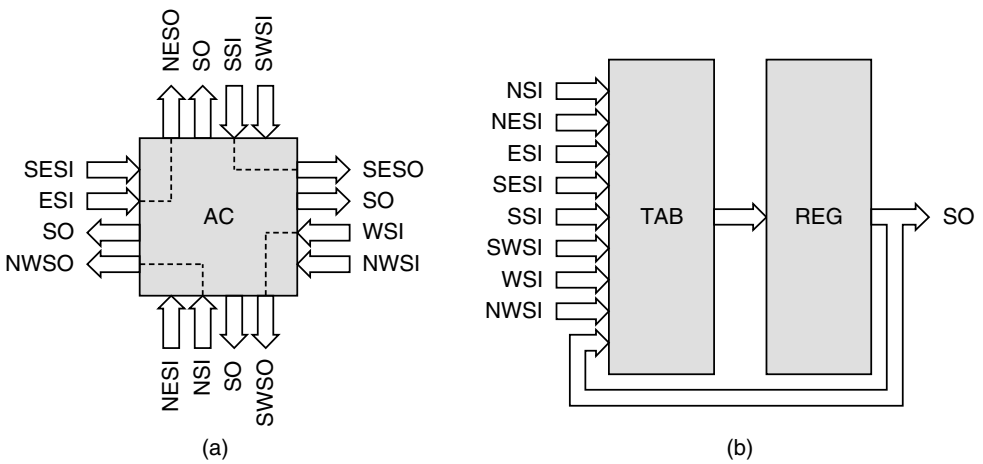


Figure 2. Two-dimensional, nine-neighbor CA. (a) Basic cell. (b) Rule table TAB and state register REG.

suitability for implementation as an actual digital system, and (2) suitability for designing growing structures.

Sections 3 and 4 introduce the growing structures of self-replicating loops and biomorphs. These sections also describe the design of their corresponding automaton cells. Their hardware implementations are discussed in Section 5. Finally, we present concluding remarks in Section 6.

The $n \times n$ self-replicating loops designed in Section 3 generalize the interactive self-replicators introduced previously [10], which were based on the traditional CA model. Essentially, we present for the first time growing structures within the novel model of the data-and-signals CA. Moreover, while the size of the state-based loops was always greater than 2 ($n > 2$), the new digital system-based loops include also the smallest possible loop ($n = 2$).

2 Data-and-Signals Cellular Automaton

The basic cell of our novel automaton, the DSCA, works with some or all of the northward (N), northeastward (NE), eastward (E), southeastward (SE), southward (S), southwestward (SW), westward (W), and northwestward (NW) directed data (D) and signals (S) (Figure 3a and 4a). The cell computes its digital outputs O from its

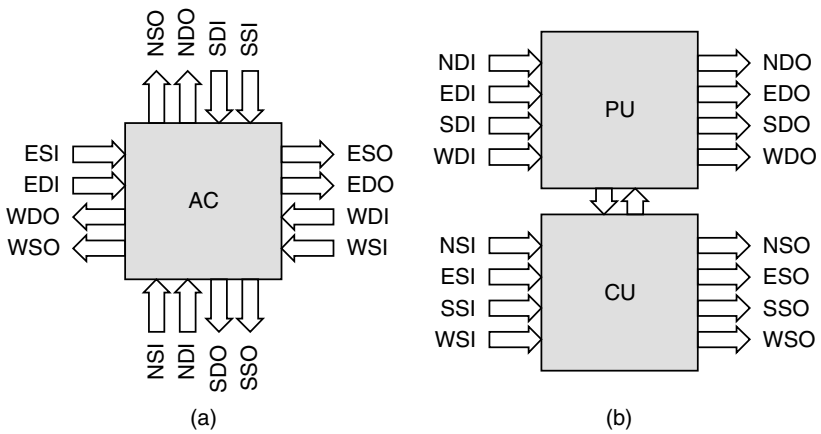


Figure 3. Two-dimensional, five-neighbor DSCA. (a) Basic cell. (b) Processing unit PU and control unit CU.

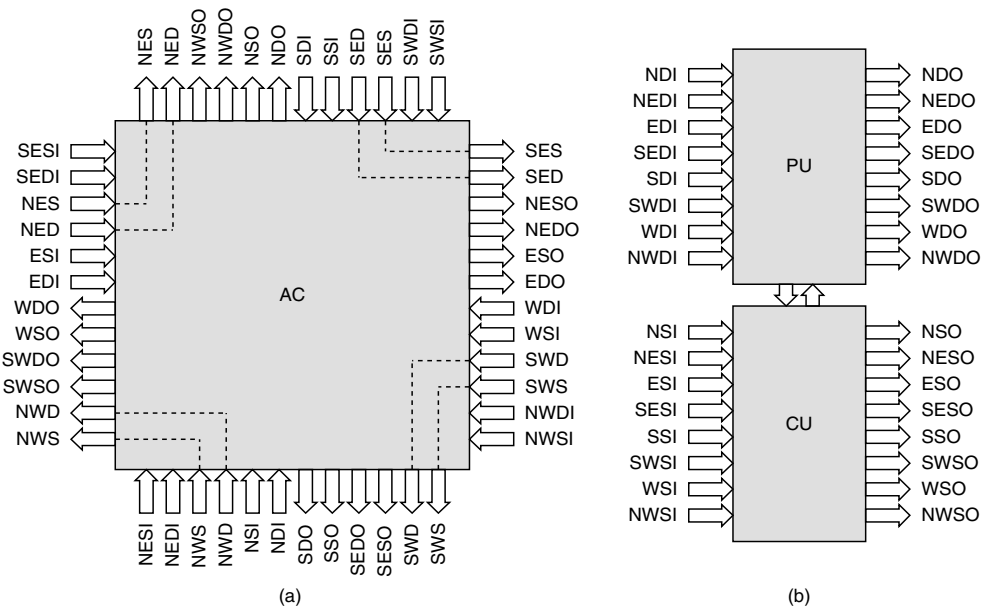


Figure 4. Two-dimensional, nine-neighbor DSCA. (a) Basic cell. (b) Processing unit PU and control unit CU.

digital inputs I. As opposed to the state shared by neighboring cells in a traditional CA (Figures 1 and 2), these data-and-signals outputs are not necessarily identical for all the neighboring cells in our new model.

In growing structures, the data and signals represent two different types of information. The *data* constitute the information that travels through the grown structure. The *signals* constitute the information that controls the growth of the structure.

Each cell of the automaton is designed as a *digital system*, resulting from the interconnection of a data-processing unit PU and a control unit CU (Figure 3b and 4b). In this digital system, the *processing unit* handles the data. It is made up of input selectors SEL, data registers REG, and output buffers BUF (Figure 5a and 6a). The *control unit* of the digital system computes the signals. It combines input encoders ENC, control registers REG, and output generators GEN (Figure 5b and 6b).

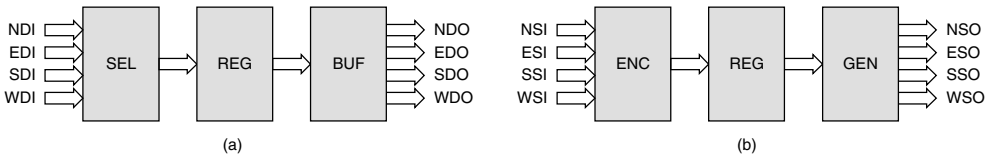


Figure 5. Detailed architecture of the five-neighbor DSCA cell. (a) Processing unit. (b) Control unit.

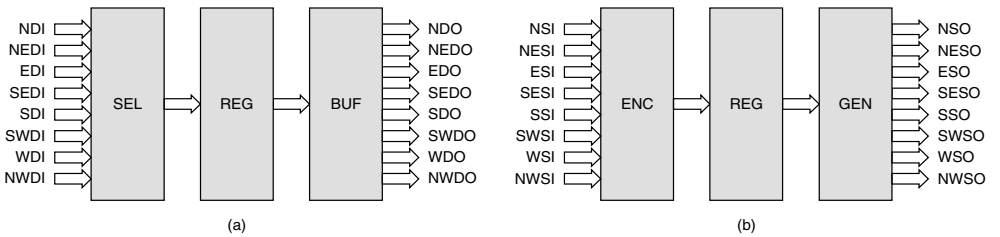


Figure 6. Detailed architecture of the nine-neighbor DSCA cell. (a) Processing unit. (b) Control unit.

Starting from the data information and the control states of a given application, the design of the cell is achieved in four steps: (1) declaration of the data and control registers, (2) description of the data and control registers in operation tables, (3) computation of the variables selecting the operations in the tables, and (4) computation of the output signals.

A DSCA can readily simulate a standard CA (since we have only *added* capabilities to the basic cell without repealing any of its functionalities), and any DSCA can be converted into a form of traditional CA by assuming all outgoing data and signals as part of a single state. Thus, the two models are equivalent in theoretical computational power. However, our interest lies in *implementation*, specifically in digital electronic media. And it is here that our DSCA shines, as we specifically designed it with hardware implementation in mind.

We shall further elaborate upon our new DSCA model by way of examples given in the following sections.

3 Five-Neighbor Application: Self-Replicating Loops

3.1 Loop Specifications

The first self-replicating structure within a traditional CA was devised by Langton in 1984 [3]. This was a configuration in the form of a loop endowed with a constructing arm and replication information (the "genome"). After 151 time steps, the original (mother) loop produced an identical daughter loop, thus performing self-replication. This 8×8 loop was implemented with a sheath around it. Later, Byl [1] proposed a simplified version of Langton's loop. More recently, Reggia et al. [5] discovered that having a sheath surrounding the loop was not essential, and that its removal led to smaller self-replicating structures. Lately, Sayama introduced a new self-replicating loop model that evolves toward different forms through variation and natural selection [6].

As opposed to the loops defined in the literature, which perform self-replication continually, the loops designed below are idle unless physically activated [9]. These unsheathed $n \times n$ loops, with $n \geq 2$, correspond therefore to interactive self-replicating loops as presented previously by us in [10]. They are implemented as growing systems involving component, apex, and data information, as shown in Figure 7.

With no external trigger, the 3×3 loop remains idle, continually producing the same eight-time-step cycle of Figure 8. When activated, the idle loop self-replicates.

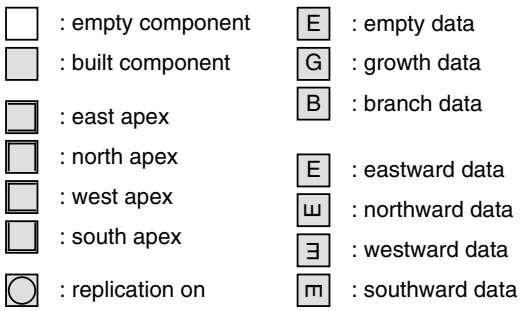


Figure 7. Data information (empty, growth, branch) and control states (component, apex, replication, direction) of the self-replicating loops.

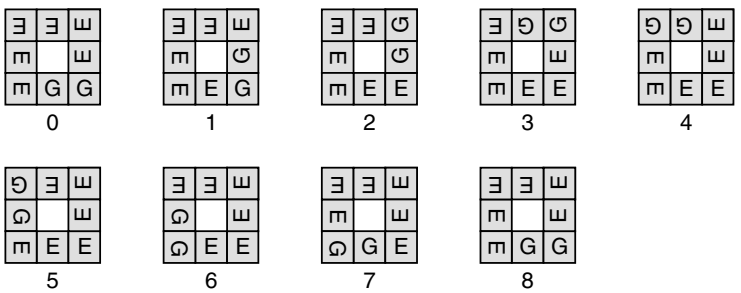


Figure 8. The eight-time-step idle cycle of the 3 × 3 loop.

Depending on the initially activated cell, the self-replication process can occur in any of the four cardinal directions. Figure 9 shows the self-replication process performed eastward. This process is executed in 48 time steps.

3.2 Loop Cell Design

The cell is a digital system whose processing unit assures the propagation of the data and whose control unit produces the signals needed for growth and cleavage of the self-replicated loop. In addition to data propagation, the digital system must perform the operations represented in Figure 10. The resources needed in order to perform the operations define the architecture of the cell (Figure 11). The processing unit involves the following subunits:

- A 2-bit data register D1:0 for the memorization of the output data $DO1{:}0$ (empty=00, growth=01, branch=10).
- A 4-input multiplexer MUX for the selection of one of the four input data lines, $EDI1{:}0$, $NDI1{:}0$, $WDI1{:}0$, or $SDI1{:}0$.

The control unit consists of six subunits:

- A 2-bit transmission register T1:0 for the memorization of the input selection (eastward=00, northward=01, westward=10, southward=11).
- A 1-bit control register B to indicate whether the component is empty ($B = 0$) or built ($B = 1$).

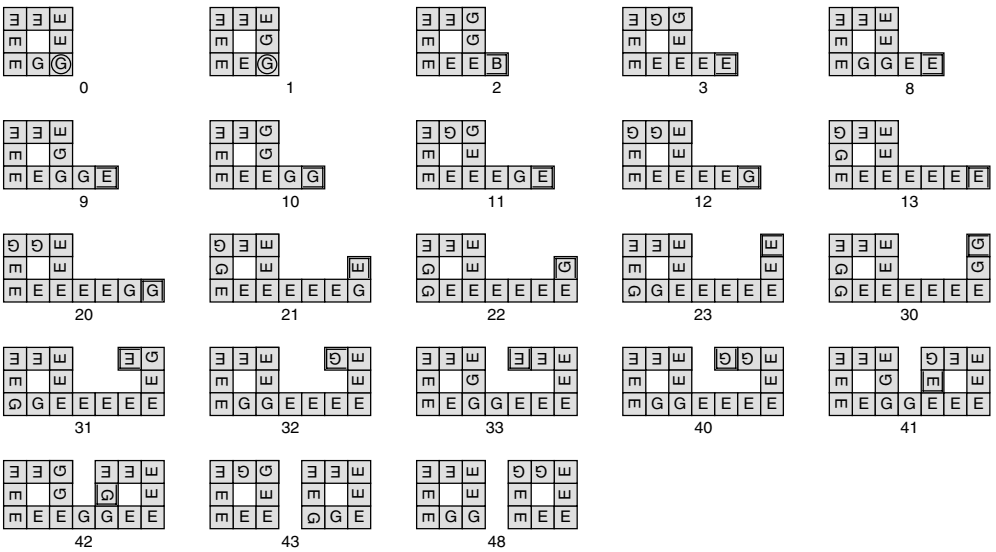


Figure 9. The eastward-directed self-replication process of the 3 × 3 loop.

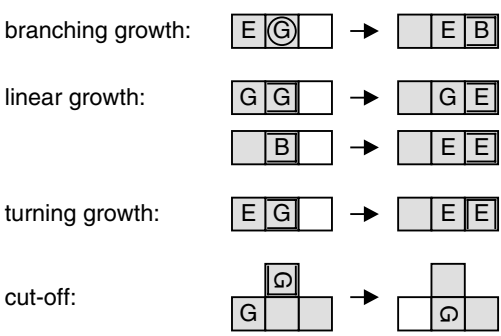


Figure 10. The growth and cleavage operations involved in the self-replication process. The data information and control states of the rightmost cell at the next time step depend on the current ones of the middle and leftmost cells.

- A 3-bit apex register A2:0 to point out the extremity and orientation of the growing structure (empty=0XX, east=100, north=101, west=110, south=111).
- A 1-bit replication register R to memorize the activation of the cell (off=0, on=1).
- An encoder ENC for input signals *SI*2:0 to command the operations performed by the data register D1:0 and the control registers T1:0, B, A2:0, and R.
- A generator GEN for output signals *SO*2:0 (empty=0XX, branching growth=100, linear growth=101, turning growth=110, cutoff=111) to produce these signals according to the information in the data register and the states of the control registers.

The operation tables of the registers as well as the logic equations of the encoder and the generator are given in Appendix A.

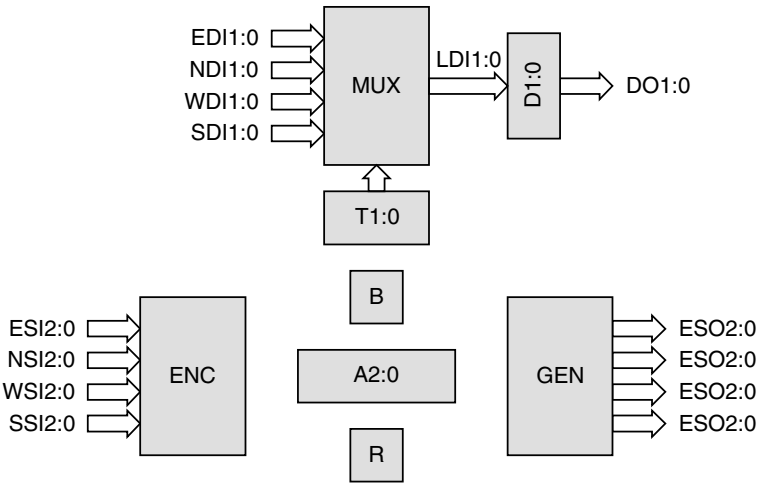


Figure 11. Detailed architecture of the self-replicating loop cell. According to the input signals *SI*, the encoder ENC defines the operations performed by the data register (D) and the control registers (T, B, A, R). According to the information and states of the registers, the generator GEN produces the output signals *SO* (see Appendix A).

4 Nine-Neighbor Application: Biomorphs

4.1 Biomorph Specifications

The term *biomorph* was coined by the surrealist artist Morris to describe animal-like shapes in his work [4]. Within the field of artificial life, Dawkins designed a computer program to explore how complex patterns (biomorphs) can arise from simple rules [2]. His main goal was to abstract and reduce to a minimum the amount of hand design required in order to design aesthetically pleasing biomorphs. Simple growing rules (embryology) and guided evolution would ideally produce biologically interesting results.

As opposed to the above biomorphs that involve mutation and evolution, the creatures described below only express a sequence of genes in order to grow (develop). They are implemented as growing structures involving the component and data information shown in Figure 12.

With no external triggering, the central component of the initial biomorph (Figure 13a) presents only empty data, and its structure remains unchanged (time step 0). When a physical input is provided, a gene G appears first (time step 1), then propagates along the legs (time step 2), and finally expresses itself in the apex components (time step 3). Depending on the number L of apex components of its initial structure (Figure 13b), the biomorph will develop a creature having four, six, or eight legs.

When activated, the central component produces a gene G with a value between 0 and 7. Figure 14a shows the expression of these eight genes for each of the eight

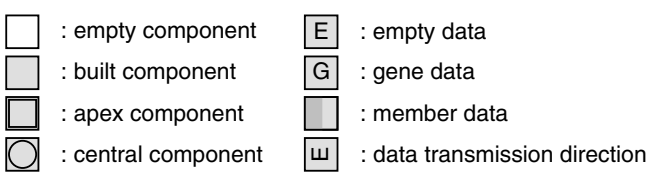


Figure 12. Data information (empty, gene, member) and control states (component, direction) of the biomorphs.

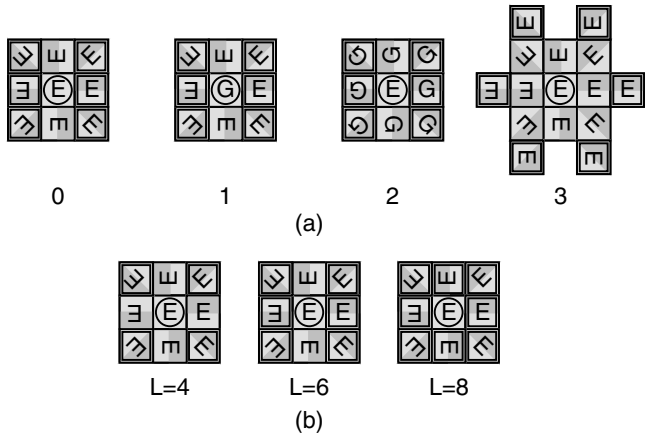


Figure 13. Biomorph. (a) Three-time-step growth cycle. (b) Four-legged, six-legged, and eight-legged creature.

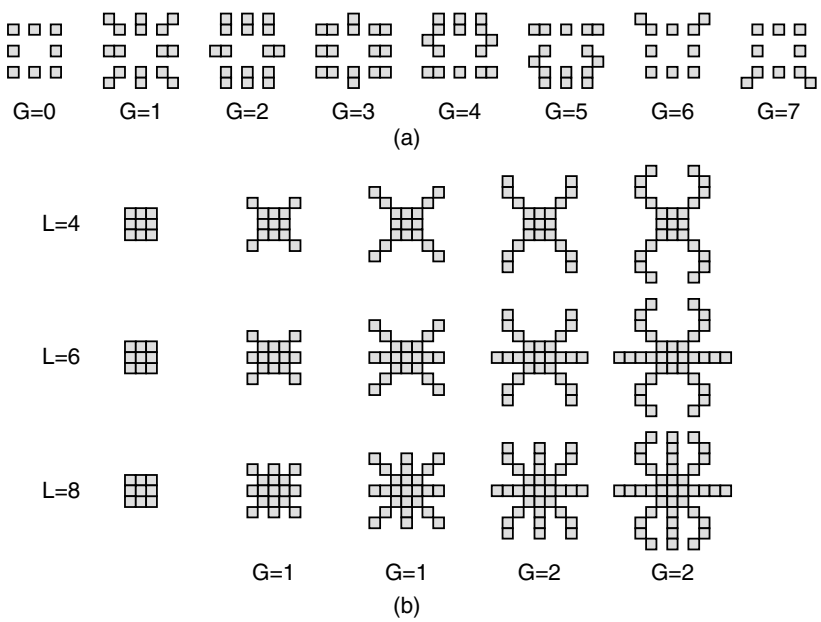


Figure 14. Genes 0 to 7. (a) Expression for all the legs of the biomorph. (b) Developments resulting from the sequence $G = 1, 1, 2, 2$.

legs of the biomorph. Starting with four-legged, six-legged, and eight-legged initial biomorphs and applying these expressions to a sequence of four genes $G = 1, 1, 2, 2$ leads to the growth developments of Figure 14b.

4.2 Biomorph Cell Design

The cell is a digital system whose processing unit assures the propagation of the data and whose control unit produces the signals required for the growth of the biomorph. In addition to the data propagation, the digital system must perform the gene expression represented in Figure 15. The resources needed in order to perform these tasks define

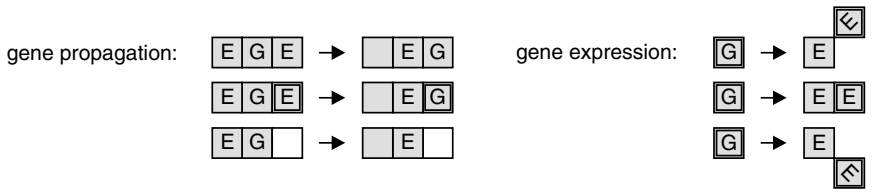


Figure 15. Propagation and expression operations involved in the growth process. The data information and control states of the rightmost cells at the next time step depend on the current ones of middle and leftmost cells.

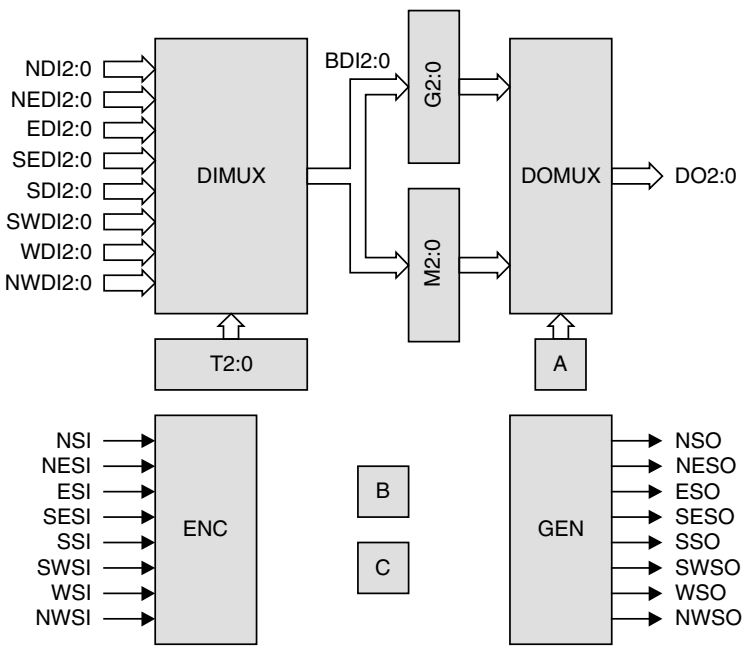


Figure 16. Detailed architecture of the biomorph cell. According to the input signals *SI*, the encoder ENC determines the operations performed by the data registers (G, M) and the control registers (T, A, B, C). According to the information and states of the registers, the generator GEN delivers the output signals *SO* (see Appendix B).

the architecture of the cell (Figure 16). The processing unit involves the following subunits:

- A 3-bit data register G2:0 for the memorization of the gene (000=empty, 001=straight growth, 010=curved in, 011=curved out, 100=curved up, 101=curved down, 110=top growth, 111=bottom growth).
- A 3-bit data register M2:0 for the memorization of the leg (000=north, 001=northeast, 010=east, 011=southeast, 100=south, 101=southwest, 110=west, 111=northwest).
- A 8-input multiplexer DIMUX for the selection of one of the eight input data $NDI2{:}0$, $NEDI2{:}0$, $EDI2{:}0$, $SEDI2{:}0$, $SDI2{:}0$, $SWDI2{:}0$, $WDI2{:}0$, or $NWDI2{:}0$.
- A 2-input multiplexer DOMUX for the selection of the gene or the leg as output data $DO2{:}0$.

The control unit consists of six subunits:

- A 3-bit transmission register T2:0 for the memorization of the input selection (000=northward, 001=northeastward, 010=eastward, 011=southeastward, 100=southward, 101=southwestward, 110=westward, 111=northwestward).
- A 1-bit control register B to indicate whether the component is empty ($B = 0$) or built ($B = 1$).
- A 1-bit apex register A to point out each extremity of the growing structure ($A = 1$) and perform the selection of its gene ($A = 0$) or its leg ($A = 1$) as output data.
- A 1-bit control register C to define the central component of the biomorph ($C = 1$).
- An encoder ENC for input signals SI to direct the operations performed by the data registers G2:0 and M2:0 as well as the control registers T2:0, B, A, and C.
- A generator GEN for output signals SO to create these signals according to the information in the data registers and the states of the control registers.

The operation table of the biomorph cell and the logic equations of its encoder and generator are given in Appendix B.

5 Hardware Implementation

The hardware implementations of the five-neighbor loop and the nine-neighbor biomorph take place in our two-dimensional electronic wall for bio-inspired systems, the *BioWall* (Figure 17) [11]. In these implementations, each cell of the automaton corresponds to a unit in the wall. This unit is the combination of three elements: (1) an input device, (2) a digital circuit, and (3) an output display.

The unit's outer surface consists of a touch-sensitive panel that acts like a digital switch. This switch enables the user to click on a corner cell of the loop or on the central cell of the biomorph. This click activates the self-replication process of the loop or generates a gene in order to activate the growth of the biomorph legs.

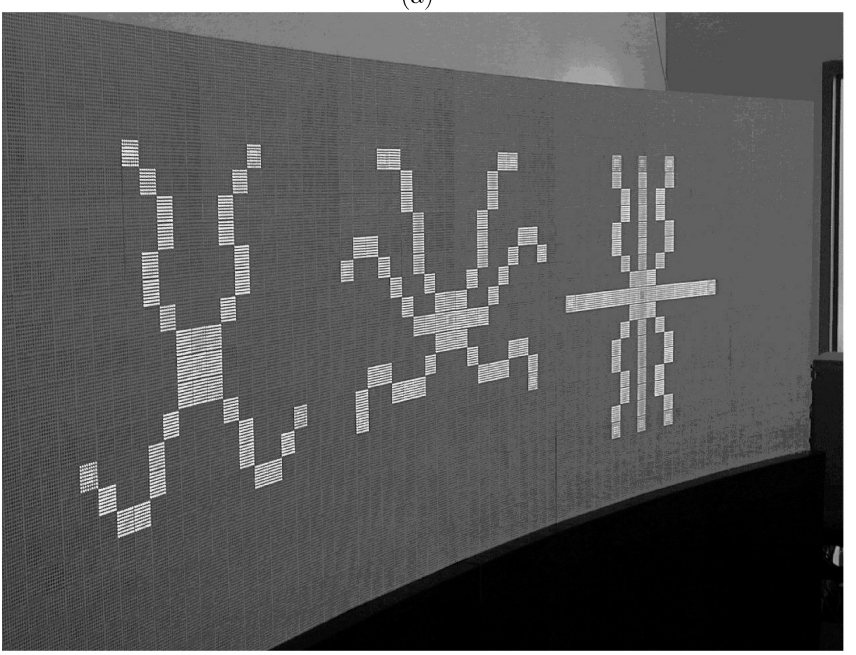
The unit's internal digital circuit is a field-programmable gate array (FPGA), configured so as to implement: (1) external (touch) input, (2) execution of the propagation and expression operations involved in the growth processes, and (3) control of the output display. This latter is a two-color light-emitting diode (LED) display, made up of 128 diodes arranged as an 8×8 dot matrix, each dot containing a green and a red LED. The display allows the user to view the cell's current data and to choose the gene generated by the central component.

6 Concluding Remarks

We have presented a novel cellular automaton dealing with data and signals instead of states. Though this automaton is especially well suited for the realization of growing structures, it constitutes a novel general CA and may be applied to other cellular applications. The DSCA greatly simplifies the design of CA involving a large number of states, which causes the rule table of a traditional CA to explode.

In the loop and biomorph applications, where only data propagation takes place when the replication or growth processes are not activated, it is quite natural to build a data path as the processing unit of a digital system. On the other side, the control unit of this digital system delivers signals only when the replication or the growth of the structures occurs.

(a)



(b)

Figure 17. The BioWall used to physically implement the growing structures. (a) Five-neighbor loop application. (b) Nine-neighbor biomorph application. (Photographs by A. Badertscher.)

The design of the data-processing unit and the control unit of the five-neighbor automaton cell in the self-replicating loop introduced the characteristic resources and registers of all digital systems. This digital-system-based cell is entirely generic, allowing the realization of all $n \times n$ loops with $n \geq 2$, contrary to the former sequential machine-based ones [9, 10], where distinct rule tables were needed, depending on the loop size. The hardware implementation of the loops takes place in the BioWall, our electronic wall for bio-inspired applications.

The biomorphs are growing structures whose complex behavior requires the design of a data-and-signals automaton instead of a state-based one. The data-processing unit and the control unit of its nine-neighbor basic cell involve the usual resources and registers of all digital systems. The hardware implementation of the biomorphs also takes place in the BioWall.

Acknowledgments

This work was supported in part by the Swiss National Science Foundation under grant 20-100049.1, by the Leenaards Foundation, Lausanne, Switzerland, and by the Villa Reuge, Ste-Croix, Switzerland.

References

1. Byl, J. (1989). Self-reproduction in small cellular automata. *Physica D, 34,* 295–299.
2. Dawkins, R. (1989). The evolution of evolvability. In C. Langton (Ed.), *Artificial life* (pp. 201–220). Santa Fe, NM: Addison-Wesley.
3. Langton, C. (1984). Self-reproduction in cellular automata. *Physica D, 10,* 135–144.
4. Morris, D. (1987). *The secret surrealist: The paintings of Desmond Morris.* Oxford, UK: Phaidon Press.
5. Reggia, J. A., Armentrout, S. L., Chou, H.-H., & Peng, Y. (1993). Simple systems that exhibit self-directed replication. *Science, 259,* 1282–1287.
6. Sayama, H. (1999). A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space. *Artificial Life, 5,* 343–365.
7. Sipper, M. (1997). *Evolution of parallel cellular machines: The cellular programming approach.* Heidelberg, Germany: Springer-Verlag.
8. Sipper, M. (2002). *Machine nature: The coming age of bio-inspired computing.* New York: McGraw-Hill.
9. Stauffer, A., & Sipper, M. (2001). Externally controllable and destructible self-replicating loops. In J. Kelemen & P. Sosik (Eds.), *Advances in artificial life: Proceedings of the 6th European Conference on Artificial Life (ECAL 2001).* Lecture Notes in Artificial Intelligence, 2159 (pp. 282–291). Heidelberg, Germany: Springer-Verlag.
10. Stauffer, A., & Sipper, M. (2002). An interactive self-replicator implemented in hardware. *Artificial Life, 8,* 175–183.
11. Tempesti, G., Mange, D., Stauffer, A., & Teuscher, C. (2002). The BioWall: An electronic tissue for prototyping bio-inspired systems. In A. Stoica, J. Lohn, R. Katz, D. Keymeulen, & R. S. Zebulum (Eds.), *Proceedings of the 2002 NASA/DOD Workshop Conference on Evolvable Hardware* (pp. 221–230). Los Alamitos, CA: IEEE Computer Society Press.
12. Toffoli, T., & Margolus, N. (1987). *Cellular automata machines.* Cambridge, MA: MIT Press.
13. von Neumann, J. (1966). *Theory of self-reproducing automata.* Urbana, IL: University of Illinois Press. Edited and completed by A. W. Burks.

Appendix A: Operation Tables and Logic Equations of the Self-Replicating Loops

Tables A1 to A5 are the individual operation tables describing the data register and the control registers of the self-replicating loop cell (Figure 11).

Table A1. Operation table of the data register D1:0 (*BSI*: branch signal input, *LDI*1:0: loop data information).

Operation	Description	B	BSI
Hold	$D \Leftarrow D$	0	0
Load	$D \Leftarrow LDI$	1	—
Branch	$D \Leftarrow 10$	0	1

Table A2. Operation table of the control register B (*GSI*: growth signal input; *CSI*: cutoff signal input).

Operation	Description	GSI	CSI
Hold	$B \Leftarrow B$	0	0
Set	$B \Leftarrow 1$	1	—
Reset	$B \Leftarrow 0$	0	1

Table A3. Operation table of the transmission register T1:0 (*VSI*: valid signal input, *TCI*1:0: transmission control information).

Operation	Description	VSI
Hold	$T \Leftarrow T$	0
Load	$T \Leftarrow TCI$	1

Table A4. Operation table of the apex register A2:0 (*ASI*: apex signal input, *VSO*: valid signal output, *ACI*2:0: apex control information).

Operation	Description	B	ASI	VSO
Hold	$A \Leftarrow A$	—	0	0
		1	—	0
Load	$A \Leftarrow ACI$	0	1	0
Reset	$A \Leftarrow 000$	—	—	1

The encoder ENC computes the control signals and data of the operation tables according to the following equations, which also include the north (*NGSI*), west (*WGSI*), and south (*SGSI*) growth signal inputs:

$$BSI = ESI2.ESI1'.ESI0' + NSI2.NSI1'.NSI0' + WSI2.WSI1'.WSI0' + SSI2.SSI1'.SSI0' \quad (1)$$

$$GSI = ESI2.ESI1' + NSI2.NSI1' + WSI2.WSI1' + SSI2.SSI1' + ESI2.ESI0' + NSI2.NSI0' + WSI2.WSI0' + SSI2.SSI0' \quad (2)$$

$$CSI = ESI2.ESI1.ESI0 + NSI2.NSI1.NSI0 + WSI2.WSI1.WSI0 + SSI2.SSI1.SSI0 \quad (3)$$

$$VSI = ESI2 + NSI2 + WSI2 + SSI2 \quad (4)$$

$$TCI1 = WSI2 + SSI2 \quad (5)$$

$$TCI0 = NSI2 + SSI2 \quad (6)$$

Table A5. Operation table of the replication register R (*EIN*: external input, *BSO*: branch signal output).

Operation	Description	*EIN*	*BSO*
Hold	$R \Leftarrow R$	0	0
Set	$R \Leftarrow 1$	1	—
Reset	$R \Leftarrow 0$	0	1

$$ASI = BSI + GSI \tag{7}$$
$$VSO = ESO2 + NSO2 + WSO2 + SSO2 \tag{8}$$
$$ACI2 = 1 \tag{9}$$
$$ACI1 = WGSI + SGSI \tag{10}$$
$$ACI0 = NGSI + SGSI \tag{11}$$
$$NGSI = NSI2 \cdot NSI1' + ESI2 \cdot ESI1 \cdot ESI0' \tag{12}$$
$$WGSI = WSI2 \cdot WSI1' + NSI2 \cdot NSI1 \cdot NSI0' \tag{13}$$
$$SGSI = SSI2 \cdot SSI1' + WSI2 \cdot WSI1 \cdot WSI0' \tag{14}$$
$$BSO = ESO2 \cdot ESO1' \cdot ESO0' + NSO2 \cdot NSO1' \cdot NSO0' + WSO2 \cdot WSO1' \cdot WSO0' + SSO2 \cdot SSO1' \cdot SSO0' \tag{15}$$

The generator GEN implements the output signals $SO2:0$. The computation of the eastward signals $ESO2:0$ involves the east branching growth EBG, east linear growth ELG, east turning growth ETG, and east cutoff ECO variables:

$$ESO2 = B \cdot (EBG + ELG + ETG + ECO) \tag{16}$$
$$ESO1 = B \cdot (ETG + ECO) \tag{17}$$
$$ESO0 = B \cdot (ELG + ECO) \tag{18}$$
$$EBG = (D1' \cdot D0) \cdot (LDI1' \cdot LDI0') \cdot (T1' \cdot T0') \cdot R \tag{19}$$
$$ELG = (D1' \cdot D0) \cdot (LDI1' \cdot LDI0) \cdot (A2 \cdot A1' \cdot A0') + (D1 \cdot D0') \cdot (A2 \cdot A1' \cdot A0') \tag{20}$$
$$ETG = (D1' \cdot D0) \cdot (LDI1' \cdot LDI0') \cdot (A2 \cdot A1' \cdot A0') \tag{21}$$
$$ECO = (T1 \cdot T0') \cdot NSI2 \tag{22}$$

Appendix B: Operation Tables and Logic Equations of the Biomorphs

Table A6 presents the global operation table describing the apex register A, the control register B, the gene register G2:0, the leg register M2:0, and the transmission register T2:0 of the biomorph cell (Figure 16).

The encoder ENC computes the control signals and the data of the operation table according to the following equations, where $SEL2:0$ are the selection variables of the input multiplexer DIMUX:

$$VSI = NSI + NESI + ESI + SESI + SSI + SWSI + WSI + NWSI \tag{23}$$
$$VSO = NSO + NESO + ESO + SESO + SSO + SWSO + WSO + NWSO \tag{24}$$
$$TCI2 = NSI' \cdot NESI' \cdot ESI' \cdot SESI' \cdot (SSI + SWSI + WSI + NWSI) \tag{25}$$
$$TCI1 = NSI' \cdot NESI' \cdot (ESI + SESI) + NSI' \cdot NESI' \cdot ESI' \cdot SESI' \cdot (WSI + NWSI) \tag{26}$$
$$TCI0 = NSI' \cdot NESI \tag{27}$$
$$SEL2 = B' \cdot TCI2 + B \cdot T2 \tag{28}$$

Table A6. Operation table of the biomorph cell (*VSI*: valid signal input; *VSO*: valid signal output; *BDI*2 : 0: biomorph data information; *TCI*2:0: transmission control information).

Operation	Description	B	A	*VSI*	*VSO*
Idle cell	$G \Leftarrow 000$	0	—	0	—
Member growth	$B \Leftarrow 1$	0	—	1	—
	$A \Leftarrow 1$				
	$G \Leftarrow 000$				
	$M \Leftarrow BDI$				
	$T \Leftarrow TCI$				
Gene propagation	$G \Leftarrow BDI$	1	0	—	—
Idle apex	$G \Leftarrow BDI$	1	1	—	0
Gene expression	$A \Leftarrow 0$	1	1	—	1
	$G \Leftarrow BDI$				

$$SEL1 = B' \cdot TCI1 + B \cdot T1 \tag{29}$$

$$SEL0 = B' \cdot TCI0 + B \cdot T0 \tag{30}$$

The generator GEN implements the output signals SO. The computation of these signals involves the external input EIN and eight memories MEM0 to MEM7 addressed by the gene, leg, and transmission registers:

$$NSO = C' \cdot B \cdot A \cdot MEM0(G, M, T) + C \cdot EIN \tag{31}$$

$$NESO = C'.B.A.MEM1(G, M, T) + C.EIN \tag{32}$$

$$ESO = C'.B.A.MEM2(G, M, T) + C.EIN \tag{33}$$

$$SESO = C'.B.A.MEM3(G, M, T) + C.EIN \tag{34}$$

$$SSO = C'.B.A.MEM4(G, M, T) + C.EIN \tag{35}$$

$$SWSO = C'.B.A.MEM5(G, M, T) + C.EIN \tag{36}$$

$$WSO = C'.B.A.MEM6(G, M, T) + C.EIN \tag{37}$$

$$NWSO = C'.B.A.MEM7(G, M, T) + C \cdot EIN \tag{38}$$