Wireworld++: A Cellular Automaton for Simulation of Nonplanar Digital Electronic Circuits

Vladislav Gladkikh
School of Information Technologies and Engineering, ADA University
11 Ahmadbay Agha-Oglu Street, Baku, AZ1008, Azerbaijan

Alexandr Nigay
Department of Computer Engineering and Telecommunications
International University of Information Technologies
Manas Str./Zhandosov Str., 34A/8A, Almaty, 050040, Kazakhstan

An enhanced version of the Wireworld cellular automaton called Wireworld++ is introduced. It can be considered as a generalization of Wireworld suitable for modeling digital electronic circuits that have intersections of unconnected wires. As most electronic circuits except trivial ones have wire crossings, Wireworld++ is a more convenient cellular automaton for modeling digital electronics than the conventional Wireworld. Wireworld++ is two dimensional; it has a small number of states and simple and intuitive rules. Despite that, it allows simulation of three-dimensional elements of digital circuits, for instance, wire crossings or electronic components placed on both sides of printed circuit boards. The key electronic parts, such as logic gates, implemented in Wireworld++ exhibit more symmetry and utilize fewer cells than their Wireworld counterparts. Wireworld++ can also be applied to simulation of computing devices in a sub-excitable, light-sensitive Belousov–Zhabotinsky medium organized in a rectangular grid of vesicles.

Keywords: Wireworld; digital electronics; wire crossings; logic gates; Belousov–Zhabotinsky reaction

1. Introduction

Wireworld is a cellular automaton that can simulate digital electronic circuits. It was invented by Silverman in 1987 (e.g., [1]) and later popularized by Dewdney [2]. As in the case of another famous cellular automaton—Conway's Game of Life [3]—complex dynamics arise in Wireworld from very simple rules and a small number of states. Like the Game of Life, Wireworld is Turing complete.

Wireworld is defined on a two-dimensional rectangular grid. Each cell can be in one of four different states: *empty*, *electron head*, *electron tail* and *conductor*. The state of each cell at the next moment of time is determined by its current state and the states of the cells in its

https://doi.org/10.25088/ComplexSystems.27.1.19

Moore neighborhood. The Moore neighborhood consists of a central cell and the eight cells that surround it (e.g., [4]). The cells change their states according to the following rules:

1. Empty cells always stay empty.
2. If a cell is an electron head, its next state will be an electron tail.
3. If a cell is an electron tail, its next state will be a conductor cell.
4. A conductor cell becomes an electron head if one or two of the neighboring cells are electron heads. If a conductor cell has fewer than one or more than two neighbors that are electron heads, it remains a conductor.

Wireworld contains all elements necessary to simulate any digital device. Data in Wireworld is represented by patterns of moving electrons. An electron consists of two cells—an electron head and an electron tail. Using different arrangements of conductor cells, it is possible to construct structures that generate data, wires for carrying data from one place to another, valves for data transformation and memory elements. Logic gates and other data processing constructions have been simulated in Wireworld. The Wireworld computer, a Turing-complete computer implemented as a cellular automaton, was designed by Moore, Owen and others between 1990 and 1992 [5]. Various other cellular automata can be built within Wireworld, for example, elementary cellular automata [6] and Langton's Ant [7]. Figure 1 shows an 8-bit multiplier by Gardner [8]. This figure as well as all other cellular automata simulations in the paper are generated in the Golly software [9, 10].

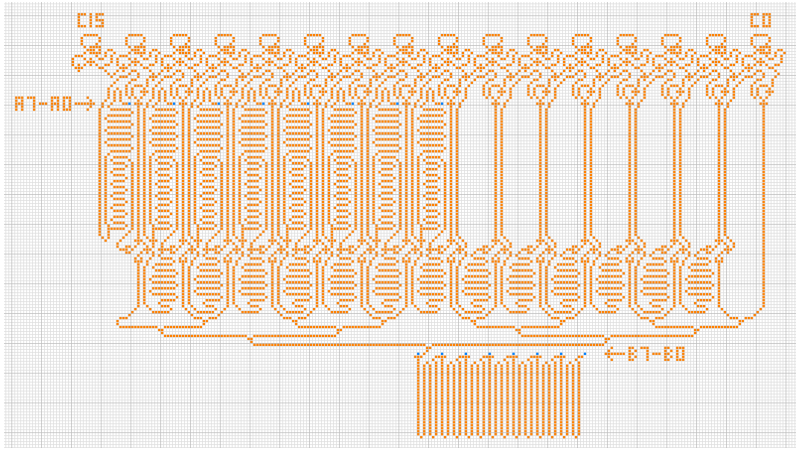

Figure 1. An 8-bit multiplier in Wireworld, designed by Gardner [8].

Since the conventional Wireworld is strictly two dimensional, it cannot simulate wire crossings in a straightforward way. The problem of wire crossings is solved in Wireworld through a workaround, by employing specific configurations of conductor cells with two inputs and two outputs. These configurations perform the function of a wire crossing, provided that some specific conditions are met for the density of electrons and the direction of their movement. This method has several drawbacks. First, such configurations do not look like intersections. This complicates the analysis of the circuit. Second, these wire crossings behave like data processing valves, not like simple wires. In particular, some of them introduce delays. Third, they employ a lot of cells. These valves are just artifacts that arise from dimensionality reduction.

To address these drawbacks, we introduce a new cellular automaton, Wireworld++. Conceptually, Wireworld++ is a generalization of the conventional Wireworld to the cases where all circuit elements do not have to lie inside a single two-dimensional circuit board. Jumper wires, multilayered conductive tracks and stacking of components on top of each other are naturally modeled in Wireworld++. In spite of the presence of these nonplanar elements, Wireworld++ is a two-dimensional cellular automaton.

Nonplanar digital circuits are ubiquitous. Figure 2 shows a simple example of a nonplanar circuit, a 4-bit barrel shifter—one of the standard combinational components of microcontrollers and microprocessors. Its nonplanarity can be proved using Kuratowski's theorem [11], which says that a finite graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 (the complete graph on five vertices) or $K_{3,3}$ (complete bipartite graph on six vertices, three of which connect to each of the other three). This circuit can be modeled by a graph shown in Figure 3 where components (multiplexers) are represented by squares, while metallic interconnections are represented by dots. The graph in Figure 3 is a complete bipartite graph $K_{4,4}$. It contains $K_{3,3}$ subgraphs, therefore it is nonplanar.

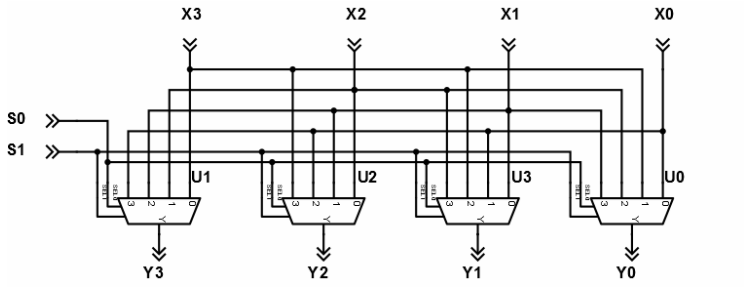

Figure 2. A 4-bit barrel shifter implemented using 4-to-1 multiplexers [12].

Multiplexers in Figure 2 are not elementary components. They are composed of switches, which nowadays are usually transistors. If we consider electronic circuits at a switch level, then we see that even simpler circuits are not planar. For example, consider an XOR gate, implemented in the CMOS logic family. The circuit is shown in Figure 4. It can be modeled by a bipartite graph in the same way as the previous example. Figure 5 shows that this graph contains a subgraph that is a subdivision of $K_{3,3}$. Therefore, this circuit is nonplanar according to Kuratowski's theorem. Modern integrated circuits may contain thousands of XOR gates as well as other, more complex nonplanar blocks. Even though the individual devices (transistors, capacitors, resistors, etc.) in integrated circuits are patterned on a single plane in a semiconductor wafer, metallic interconnections between them are organized in several layers isolated from each other by insulating layers [13, 14].

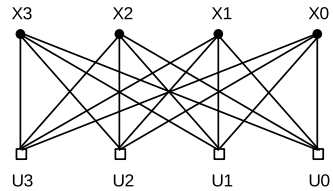

Figure 3. A graph that models the circuit from Figure 2. Components (multiplexers) are represented by squares. Metallic interconnections are represented by dots.

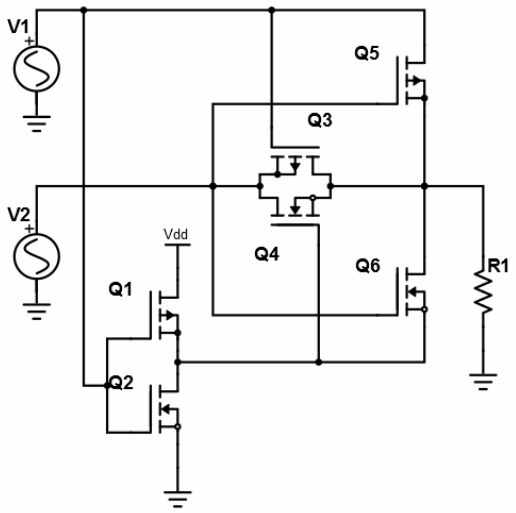

Figure 4. A CMOS XOR gate [12].

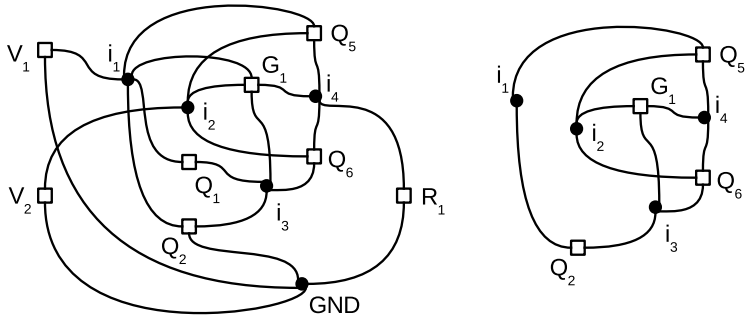

Figure 5. A graph that models: (a) the XOR gate from Figure 4; and (b) its subgraph that is a subdivision of $K_{3,3}$.

2. Uniform Streams of Electrons

In this section, we outline a few concepts required for the rest of the paper.

A *wire* is a sequence of conductor cells such that any pattern of electrons is preserved while moving along this sequence. A wire can be used for carrying data from the place where the data is generated to the place where it is transformed.

A stream of Wireworld electrons moving along a wire is called a *uniform stream* if it consists of electrons separated by the same distance. Such a stream can, for example, be generated by a clock, which is usually a circular structure with one or more electrons circling around it. Clocks have other forms as well. The period of a uniform stream of electrons is calculated as the number of cells between two subsequent electron heads in the stream (excluding the heads themselves) plus one. Because each electron occupies two cells, and at least one gap is necessary between successive electrons in a stream, the smallest possible period is equal to 3. Figure 6 shows uniform streams of electrons of periods 3, 4 and 6, each generated by its own clock.

For simplicity, we refer to a uniform stream of a period n as a *signal of period n*. Different authors use different terms for the same concept. Scherer calls it "n-cycle data." Moore and Owen call it "n-micron signal" [5]. Heise uses the term n-tick to denote the period n [7]. We prefer the term "a signal of period n," because n quantifies the signal in relation to both time and space. The number n specifies the time period of the signal, because exactly n cellular automaton generations pass between the time moments when a particular cell is in the electron head state. If one considers the stream of Wireworld electrons as a propagating wave, then the period of the wave is equal to n. Here, electron heads play the role of crests or troughs.

Alternatively, the number n is equal to the wavelength of this wave measured in the number of cells. This shows that the same n determines the period of the signal in space.

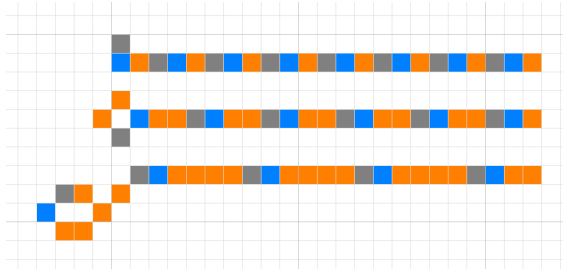

Figure 6. Signals of period 3, 4 and 6 (from top to bottom). The electron head is blue; the electron tail is gray.

A signal of period n can also be thought of as direct current. In this point of view, n is inversely proportional to the rate of the current. The less n is, the closer the electrons are to each other and the more electrons pass through a conductor cell per unit time, which means the higher the current rate.

Moore, Owen and coworkers built their Wireworld computer using logic gates that work with signals of period 6 [5]. In his original paper, Dewdney suggested processing signals of period 13 [2].

Signals of small periods tend to require valves that include large arrangements of conductor cells (see e.g., [15] for comparison of Wireworld processing valves for different n). This is undesirable, because the larger the Wireworld circuit is, the more possibilities for errors, especially in its implementation in unconventional hardware, such as in chemical substances. One of the goals in designing Wireworld circuits is to use a minimum number of cells that fulfill the desired function. Another design goal is minimization of delays in signal propagation. Delays may occur when the signal goes through a processing valve, especially if the valve is complicated. Delays can also be introduced intentionally in order to synchronize two or more signals that are out of phase. In any case, delays are undesirable because they slow down processing and increase the number of conductor cells. We took into account these goals when designing our Wireworld++.

3. Wireworld++

The cellular automaton that we introduce in this paper, Wireworld++, was inspired by the existence and ubiquity of electronic circuits that have intersections of unconnected wires. Even though electronic

circuits are sometimes considered to be two dimensional, assembled on planar printed circuit boards, in reality, only the simplest circuits are strictly two dimensional. A printed circuit board for any nontrivial electronic device has either jumper wires or conductive tracks on both sides. More complicated devices require multilayer boards. All digital circuits can be modeled by Wireworld. However, our enhanced version, Wireworld++, models wire intersections and processing valves more intuitively and makes them simpler in appearance, as is shown in the subsequent sections.

Like the conventional Wireworld, Wireworld++ is defined on a two-dimensional rectangular grid. Being two dimensional, it nevertheless can describe three-dimensional elements, like wire crossings or multiple components placed on top of each other. Using three-dimensional automata for these constructions is possible, but would result in computational complexity not justified by the relatively small number of the points where the third dimension is used. Instead, we decided to stay in two dimensions but introduce a few extra states and rules that are used only in those places where we need the third dimension. A similar approach and justification was employed in the work by Feijs in which he designed two-dimensional cellular automata for simulating analog electronics [16].

The state space of Wireworld++ consists of the following states:

1. Empty
2. Strong head
3. Strong tail
4. Strong conductor
5. Weak head
6. Weak tail
7. Weak conductor

The first four states in this list are analogous to the states of the conventional Wireworld. This is also true for states (1), (5), (6) and (7). Any Wireworld pattern can be constructed using only states (1) through (4), where the empty state and the strong conductor state correspond to Wireworld's empty and conductor states, while the strong head and the strong tail states correspond to Wireworld's electron head and tail. Alternatively, any Wireworld pattern can be created using only states (1), (5), (6) and (7). Here again, state (1) can be used as Wireworld's empty state, state (5) as Wireworld's electron head, state (6) as the electron tail and (7) as the conductor. In order to model intersections of unconnected wires, both weak and strong states are required, which will be detailed in the next section.

https://doi.org/10.25088/ComplexSystems.27.1.19

The transition rules of Wireworld++ are as follows:

1. Empty cells stay empty.
2. A strong conductor cell becomes a strong head if one or two of its neighboring cells are strong heads.
3. A strong conductor cell becomes a strong head if exactly two of its neighboring cells are weak heads.
4. A strong head becomes a strong tail.
5. A strong tail becomes a strong conductor.
6. A weak conductor becomes a weak head if one or two of its neighboring cells are weak heads.
7. A weak conductor becomes a weak head if exactly one neighboring cell is a strong head.
8. A weak head becomes a weak tail.
9. A weak tail becomes a weak conductor.

Rules (1), (2), (4) and (5) are those of conventional Wireworld applied to the strong states. Rules (1), (6), (8) and (9) are the conventional Wireworld rules for the weak states. We see that any Wireworld++ pattern that contains only states of the same strength (only weak or only strong) is a conventional Wireworld pattern. Therefore, Wireworld is a special case of Wireworld++. These rules imply that only strong signal carriers can travel along strong wires, and only weak carriers along weak wires.

Rules (3) and (7) refer to the interactions of strong and weak states. These rules are not symmetric. A strong conductor cell requires two weak heads to become a strong head, while just one strong head is sufficient for a weak conductor cell to become a weak head. This clarifies the terms "strong" and "weak." The weak head can be thought of as two times "weaker" than the strong head, because two of them are required to invoke a signal in a strong conductor. This asymmetry turns out to be important for modeling three-dimensional circuits by two-dimensional cellular automata.

Wireworld++ can be thought of as the result of some algebraic operation (let us denote it $*$) on two conventional Wireworlds:

$$W_{++} = W * W. \tag{1}$$

This operation takes the states and the rules of its operands and builds new states and new rules from them. Depending on the operation $*$ and its operands, the result may have duplicate states and rules, so only one instance in each duplicate should be retained. In our case, such a duplicate state is the empty state, and a duplicate rule is the rule that says that the empty state never changes. The result of this

operation may also have additional rules that specify what the next state should be when the states from different operands are in the same neighborhood. In our case, these are rules (3) and (7). It seems natural that Wireworld++ is built only from Wireworlds and no other cellular automata, because the same physical system is simulated in both wires and intersections of unconnected wires. The intersection should not add any new physics; it is just two wires, one above the other, which is reflected in equation (1). The same line of thought can be applied not only to intersections of unconnected wires, but also to electronic components (diodes, transistors, logic gates, multiplexers) placed on top of each other, for example, on both sides of a printed circuit board. Theoretically, this could be generalized to intersections of more than two wires at the same point, in which case the resulting cellular automaton would be

$$W_{++} = W * W * \cdots * W. \tag{2}$$

However, the number of states and rules for such an automaton would be prohibitively high, and it would not bring any advantage in simulation of digital circuits, as intersections of more than two wires are extremely rare. Even if they are present in a certain circuit, they can be modeled as intersections of pairs of wires in close but separate points. Furthermore, it turns out to be possible to implement multiple wire crossings at one point in Wireworld++ constructed according to equation (1), as we will show in the next section. This further diminishes the necessity of the complicated automaton equation (2).

We decided not to use the word "electron" in the names of the Wireworld++ states. This is because even in the case of electric current, charge is not always carried by moving electrons. In some substances (such as electrolytes or plasma), charge is carried by moving ions. Moreover, the design or analysis of electronic circuits rarely requires studying the flow of electrons. In most cases, it is sufficient to work in terms of voltages and currents described by Kirchhoff's laws, Ohm's law and a few other theorems, without going to lower levels of physical abstractions [16]. There is also research on modeling digital circuits by cellular automata related to implementations of computing devices in chemical, biological or other systems that are called "unconventional" by computer scientists to contrast them with conventional electronics based on semiconductor technology (e.g., [17, 18]). Signals in those systems do not have to be carried by electrons. They can be transferred by chemical reactions or by traveling localized excitations, like in a light-sensitive Belousov–Zhabotinsky (BZ) medium (e.g., [19]). What is important for a charge carrier is that it moves like a free particle in empty space. When such a carrier is modeled by some totalistic, isotropic cellular automaton, at least two cells (head and tail) are required to represent it moving in a certain

https://doi.org/10.25088/ComplexSystems.27.1.19

direction; the direction of the movement can be specified by a directed line segment starting from the cell that is in the "head" state and ending in the cell that is in the "tail" state. Even though we occasionally use the word "electron" in this paper for simplicity to denote a signal carrier, we omit it in the formal definitions and rules. It is possible to simulate a movement of the signal carrier by a single cell if we employ nontotalistic and nonisotropic rules that take into account the exact relative positions of the neighbors (e.g., [20]). Such a rule can be, for example, "the cell changes to a carrier state if the cell to the left is a carrier." However, in this paper we work on a generalization of Wireworld that is isotropic and totalistic, so we decided to stay in the same class of cellular automata.

4. Wire Crossings

In classical electronic design, when two unconnected wires cross, we assume that currents can in principle flow along these wires in any direction. We also assume that the rate of current can have any value, and that the currents in different wires do not influence each other. These assumptions may not always hold in actual electronic circuits, but attempts are being made to minimize parasitic influence of wires on each other, and the most appropriate wire thickness is usually chosen for a specific range of currents.

When wire crossings are simulated by cellular automata, additional complications arise, because wire crossings are modeled by valves that may not have all the properties of a bunch of nonintersecting conductors. Some of these valves can model wire crossings only when the current goes in one particular direction. Other valves model wire crossings when electrons flow only along one of the wires, but not when they enter the intersection simultaneously from two different wires.

A Wireworld wire crossing is called *unidirectional* if it allows data to pass correctly only in one direction. The crossing may not work correctly when data is sent from the other direction. A Wireworld wire crossing is called *bidirectional* if it correctly passes the data in both directions. A Wireworld wire crossing is called *single channel* if only one stream of data can pass through it at any given moment of time. A Wireworld wire crossing is called *double channel* if two streams of electrons can pass through it simultaneously.

The asymmetry of the Wireworld++ rules allows constructing a wire crossing as shown in Figure 7. Throughout the paper, we use lighter colors for the strong states and darker colors for the weak states. The wires consist only of the strong conductor cells. The wire crossing contains only weak conductor cells. That means that we need

weak cells in those places where the circuit needs the third dimension, as in the case of wire crossings.

Figure 8 shows the successive snapshots of a single electron passing through the intersection. Upon entering the intersection, the strong head creates two weak heads. These two weak signal carriers move through the intersection together and produce a strong head in the first strong conductor cell after the intersection, according to rule (3). The weak heads do not affect the wires on the sides because only one weak head touches each wire, whereas, according to rule (3), two weak heads are required for a strong conductor cell to become a strong head.

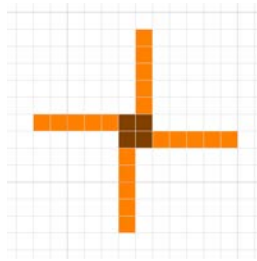

Figure 7. A bidirectional, single-channel wire crossing for signals of period $n \geq 4$.

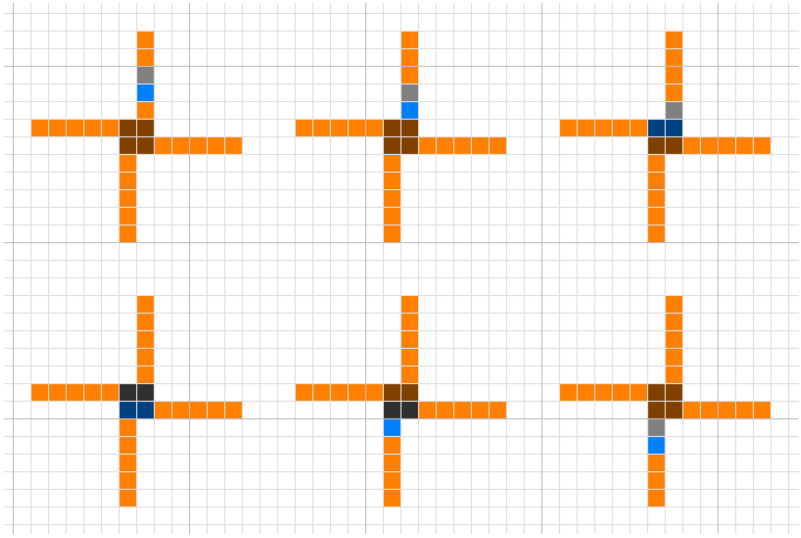

Figure 8. The successive snapshots of a single electron passing through the wire crossing (from left to right, top to bottom).

https://doi.org/10.25088/ComplexSystems.27.1.19

This wire crossing does not introduce any delay compared to a straight wire. It is a single-channel, bidirectional crossing that works for signals of period $n \geq 4$. To the best of our knowledge, there is no wire crossing in the conventional Wireworld that is so simple and looks so much like a real intersection of unconnected wires. The closest Wireworld analog that we could find is a wire crossing shown in Figure 9 that works for signals of period $n > 4$ [15]. However, this wire crossing is unidirectional. It allows signals to pass correctly only from left to right. Moreover, it contains more cells than our Wireworld++ construction, and it looks like a processing valve rather than a wire crossing.

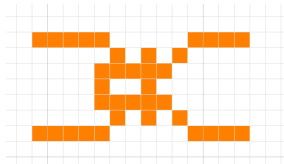

Figure 9. A unidirectional Wireworld wire crossing for signals of period $n > 4$.

The wire crossing in Figure 7 is single channel, so if two signals are sent simultaneously toward the intersection along the perpendicular wires, they will disappear at the square of weak cells. Wireworld electrons also disappear if they travel in the same wire toward each other and collide. In case two signals arrive simultaneously from two opposite directions toward this wire crossing, they will exit the square of weak cells along the two other wires perpendicular to the incoming wires. Because of such behavior, this intersection can be thought of as two Γ-shaped insulated wires placed close to each other. It is an interesting feature of the cellular automata that the same configuration of cells can have different interpretations and serve different purposes, depending on the dynamics around it.

Figure 10 shows why this wire crossing does not work for signals of period three but works for signals of period greater than or equal to four. Let us send the alternating sequence 101 010… along one wire and the sequence 010 101… along another wire. The wire crossing is single channel, but in this case the electrons do not enter the wire crossing simultaneously, so it is expected to work properly. When a logic one enters the wire crossing from one wire, a logic zero enters it from another wire, and vice versa. In Figure 10, we see that when the period of the signals is three, both signals change after the wire crossing, while they pass undisturbed when their period is four. The reason is that the square of weak cells can handle correctly only one signal; thus the period should be large enough for both signals to pass without interference.

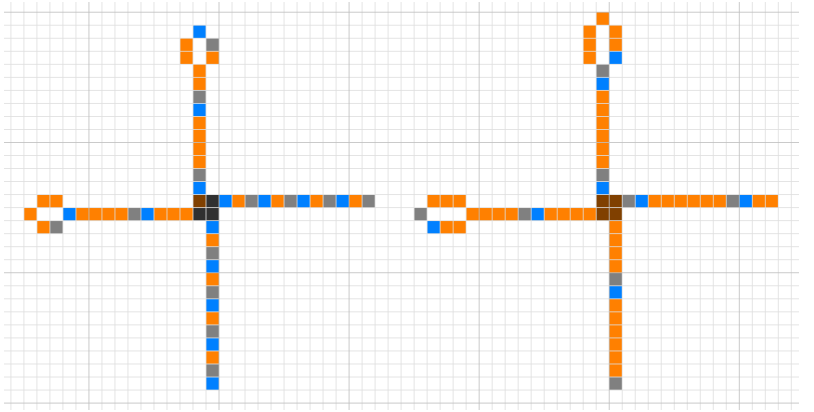

Figure 10. The signal 101 010... travels along one wire, and the signal 010 101... travels along the perpendicular wire. The signals in the configuration on the left have period $n = 3$. They cannot pass through the intersection correctly. The signals in the configuration on the right have period $n = 4$. The wire crossing works perfectly for them.

We can now build wire crossings that work only for signals whose period is not less than a specified number n. For example, Figure 11 shows two wire crossings; one of them (on the left) works only for signals of period $n \geq 5$, while the other one (on the right) works only for signals of period $n \geq 6$. Both wire crossings are bidirectional, single channel. It is straightforward to generalize this pattern for making wire crossings that work only for larger periods. One just has to increase the perimeter of the central figure made of weak cells. These patterns will be used later in this paper as building blocks for making other constructs. We see that creating wire crossings for signals of different periods follows a well-defined pattern in Wireworld++. We have not seen in the literature any such patterns in conventional Wireworld. Wire crossings in Wireworld are completely different for each period n. We do not claim that it is impossible to find such a pattern in Wireworld, but we doubt that it will be as simple and elegant as in Wireworld++.

The passage of time for these crossings depends on their size. The crossing on the left-hand side of Figure 11 has zero delay compared to the straight wire, while the crossing on the right has a delay equal to one time cycle. Larger crossings have a delay equal to $n - 5$, where n is the smallest period of the signal that the crossing allows to pass without errors. If two signals are sent simultaneously toward any of these constructions along the perpendicular wires, they will disappear at the central shape of weak cells. In case two signals enter simultaneously from two opposite directions, they will exit the

intersection along the two other wires perpendicular to the incoming wires.

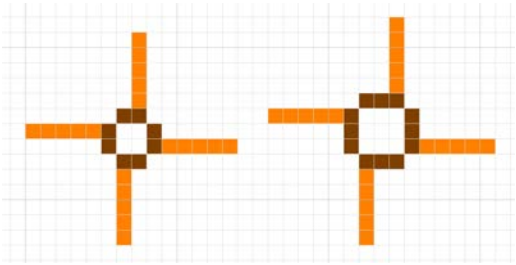

Figure 11. Wire crossings that work only for signals whose period is not less than a specified number n. On the left is a wire crossing that works only for signals of period $n \geq 5$. On the right is a wire crossing that works only for the signals of period $n \geq 6$. The generalization for larger n is straightforward.

As an example of modeling multiple unconnected wires that cross in one point, consider a single channel, bidirectional wire crossing in Figure 12. A signal can enter any wire and will exit the wire on the opposite side of the crossing. This wire crossing employs the same asymmetry of rules for strong and weak states as all the previous wire crossings: a strong head appears only when two weak heads collide, which happens at the cell exactly opposite to the entrance of the signal into the crossing. This wire crossing has no delay compared to a straight wire for signals traveling along the vertical or horizontal wire, but has a delay of three time cycles for signals traveling along the diagonal wires. As we mentioned in the previous section, Wireworld++ constructed algebraically from two instances of Wireworld is sufficient for describing even the situations where more than two unconnected wires cross at one point.

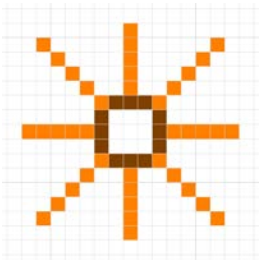

Figure 12. Four unconnected wires cross at one point. This wire crossing is single channel, bidirectional.

The construction in Figure 12 has interesting behavior when signals come to it in more than one wire simultaneously. Some examples are given in Table 1. In some of those cases, it behaves like a wire crossing; in others it does not.

Number of Entering Signals	Directions of Entering Signals	Number of Exiting Signals	Directions of Exiting Signals
2	From the opposite sides toward each other.	2	Along the wires perpendicular to the entering signals.
2	At 90° to each other.	2	One along the wire that is between the incoming wires; the other one along the opposite wire at 135° to the incoming wires.
2	At 45° to each other (neighboring wires).	1	The signal that travels in the diagonal wire disappears; the remaining signal exits on the opposite side.
2	At 135° to each other.	2	At 135° to each other but along different wires.
3 or 5 or 7	Neighboring wires.	1	Opposite to the wire that is central to the cluster of incoming signals.
4	Neighboring wires.	1	Along a diagonal wire that is opposite to the diagonal incoming signal that is between horizontal and vertical incoming signals.
6	Neighboring wires.	1	Along the remaining nondiagonal wire.
8	All wires.	0	—

Table 1. The behavior of the intersection in Figure 12 when signals come to it in more than one wire simultaneously.

The wire crossings discussed so far were single channel. Only one signal could travel through them at a time. We can take previous constructs as building blocks and create double-channel wire crossings. In Figure 13, we present a Wireworld++ double-channel wire crossing (on the left), and compare it with its Wireworld analog (on the right) designed by Scherer. Both wire crossings are unidirectional, work correctly for signals of period $n \geq 5$, and do not introduce any delay compared to a straight wire. The Wireworld++ wire crossing has half the conductor cells of its Wireworld counterpart (26 versus 52). Also, the Wireworld++ version looks more like a wire crossing, while the

https://doi.org/10.25088/ComplexSystems.27.1.19

Wireworld version looks like a valve. The directional asymmetry of the crossing (the fact that it is unidirectional) is more visible in the Wireworld++ case. Generalization to signals of larger periods is straightforward and shown in Figure 14. The only difference between the wire crossings in Figure 14 is the perimeter of the central figure made of weak cells, which is uniquely determined by the smallest period of the signal that we allow to pass through the crossing.

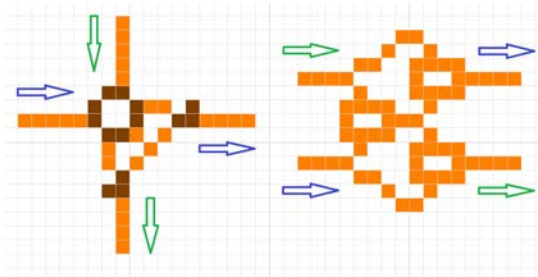

Figure 13. A Wireworld++ double-channel wire crossing (on the left), and an analogous Wireworld intersection (on the right) designed by Scherer. Both wire crossings are unidirectional and work correctly for signals of period $n \geq 5$. The arrows of different colors show the paths of each signal.

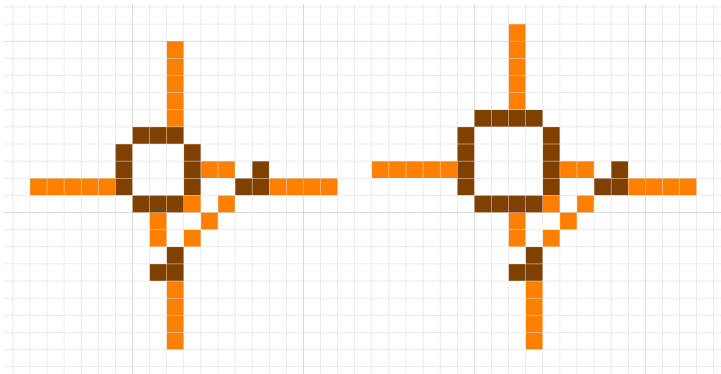

Figure 14. Double-channel wire crossings that work for signals whose period is not less than a specified number n. The smallest period n is uniquely determined by the perimeter of the central arrangements of weak cells.

Finally, we would like to present a double-channel, bidirectional wire crossing. It is shown in Figure 15 and is valid for signals of period $n \geq 6$. When only one signal passes though this wire crossing

in any of the four possible directions, it has no delay compared to a straight wire. If two signals entering from wires perpendicular to each other cross it simultaneously, they are delayed one time cycle.

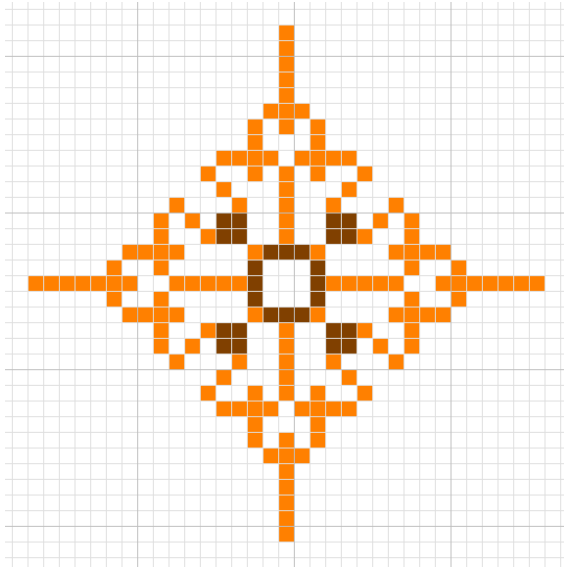

Figure 15. A double-channel, bidirectional wire crossing for signals of period $n \geq 6$.

5. Diodes and Logic Gates

A Wireworld++ diode is shown in Figure 16, where it is compared with a Wireworld diode.

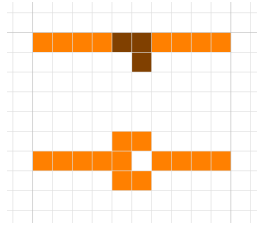

Figure 16. A Wireworld++ diode (top) and a Wireworld diode (bottom).

Figure 17 shows a Wireworld++ AND gate for signals of period $n \geq 3$ on the left and a Wireworld AND gate on the right designed by Scherer [15]. The Wireworld++ version is a winner in all respects. It is

simple, clear to understand and has no delay. Moreover, the Wireworld++ version is constructed only of conductor cells, while the Wireworld one has to have complicated internal dynamics of circulating electrons, including a constant supply of electrons entering it from the opposite direction (see the rightmost electron in the figure), which makes it practically unusable. A more popular Wireworld AND gate is shown in Figure 18. It works for signals of period $n \geq 5$. Even this simpler version uses more cells than the Wireworld++ AND gate. The Wireworld OR gate is as simple as a gate can be, so there is no need to invent an alternative version here. Wireworld NOT gates are also simple. Each NOT gate is specific for a particular period of the signal. It consists of a clock from Figure 6 producing a signal of period n and an interfering input that quenches this signal. These interfering patterns are different in Wireworld and Wireworld++ but Wireworld++ versions are not much simpler. Just for reference, we show NOT gates for $n = 4$ and $n = 6$ for both cellular automata in Figure 19.

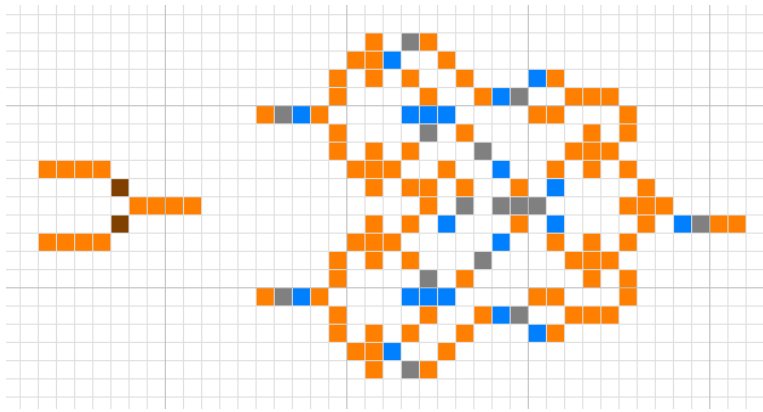

Figure 17. A Wireworld++ AND gate (left) and a Wireworld AND gate (right) for signals of period $n \geq 3$.

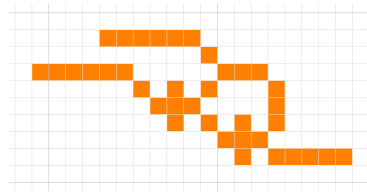

Figure 18. A Wireworld AND gate for signals of period $n \geq 5$.

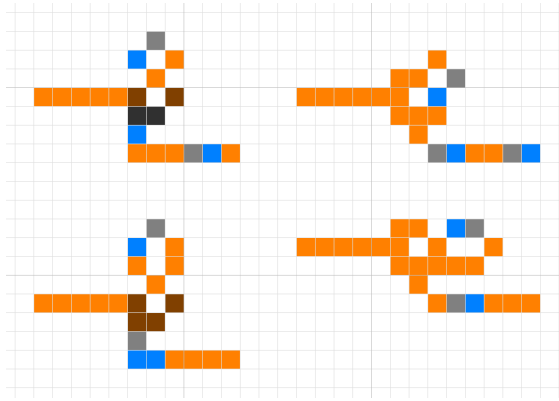

Figure 19. Wireworld++ NOT gates (left) and Wireworld NOT gates (right). The gates in the top row are for signals of period $n = 4$. The gates in the bottom row are for signals of period $n = 6$.

Figure 20 shows a Wireworld++ XOR gate for $n \geq 3$ on the left together with two Wireworld XOR gates. The gate in the center, designed by Heise, works for $n \geq 3$ [7], while the gate on the right, designed by Walraet [21], works for $n \geq 5$. The Wireworld++ XOR gate uses fewer cells than all its Wireworld counterparts.

Using these logic gates, any digital device can be simulated.

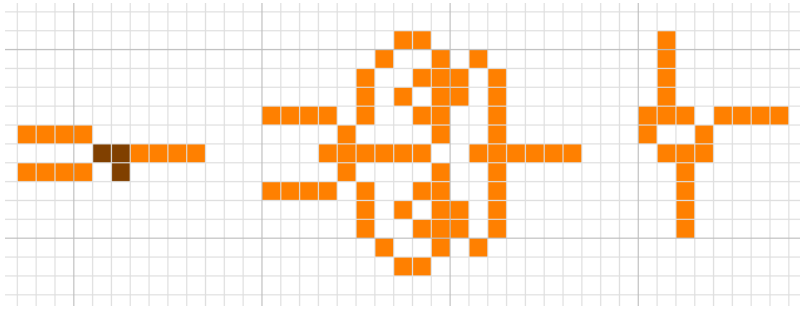

Figure 20. A Wireworld++ XOR gate for signals of period $n \geq 3$ (left) and Wireworld XOR gates. The gate in the center works for $n \geq 3$, while the gate on the right works for $n \geq 5$.

https://doi.org/10.25088/ComplexSystems.27.1.19

6. Similarities between Wireworld++ and Interacting Wave Fragments of Belousov–Zhabotinsky Oscillating Reaction

There are certain similarities between logic gates simulated in Wireworld++ and the corresponding gates implemented by colliding wave fragments in a sub-excitable, light-sensitive BZ medium. The BZ reaction is an oscillating chemical reaction [22, 23]. Under a specific, narrow range of illumination, perturbations of the ruthenium-catalyzed BZ medium lead to the formation of traveling wave fragments [24]. These wave fragments preserve their shapes and velocity for some time, so they behave like quasi-particles [25, 26]. These traveling quasi-particles can be used as signal carriers, and the results of their collisions can be interpreted as computations [27–29]. Traveling wave fragments are unstable, however. They either collapse or expand after a short time. One approach to mitigating this instability is to divide the reacting medium into compartments so small that a wave packet is stable during the time it travels across the compartment. These compartments are called BZ vesicles [30, 31]. Each BZ vesicle is enclosed in a membrane impassable for wave packets. A pore between two BZ vesicles is formed at the place where they are in contact with each other. The diameter of the pore should be such that the wave fragment, entering into the vesicle, is stable while traveling across the vesicle along a straight line. While inside the vesicle, the wave fragment may collide with other wave fragments that entered through other pores. The result of this collision may be one or more wave fragments that exit the vesicle through yet other pores. Arranging BZ vesicles in specific configurations, it is possible to build logic gates and more complex computing devices. There are many possible ways to implement a particular gate. All vesicles can either be of the same size or they can be of different sizes. They can be arranged either in a regular or an irregular grid. The pore efficiency may either be the same for all vesicles or different for each pair of vesicles. Here, we compare Wireworld++ logic gates with the gates made of orthogonal arrangements of uniform-sized BZ vesicles [32]. We notice that both BZ and Wireworld++ gates are constructed using the same ideas. Consider the comparison of the NOT gates implemented in each of the two systems in Figure 21. The BZ NOT gate consists of a constant source of excitations (permanent logic one) in the topmost disk. This is analogous to the clock (also a signal source) in Wireworld++. The central vesicle, in which the reaction takes place, corresponds to the central arrangement of weak cells in the Wireworld++ version. When the input is zero (the left-hand side of Figure 21), the output is one. When the input is one (the right-hand side of Figure 21), then both in the BZ and Wireworld++ cases this input collides with the permanent source signal, and the result of this collision misses the downward

passage to the exit. Instead, the resulting signal carrier hits the wall of the vesicle in the BZ case or a dead end in the Wireworld++ case at 45 degrees from the exit and disappears.

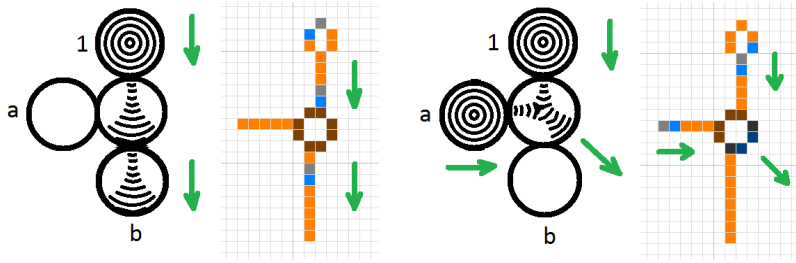

Figure 21. Similarities between implementations of a NOT gate in the BZ reaction [32] and in Wireworld++.

Let us now consider a NAND gate. It is shown in Figure 22 for both the BZ reaction and Wireworld++. The gates in both systems are again in complete analogy with each other. When at least one of the inputs *a* or *b* in the upper row of the vesicles is zero, then the signal from the permanent source on the right passes unperturbed toward the output *c*. When both *a* and *b* are ones, then the signals collide in the upper central cell. The result of this collision propagates downward and collides with the permanent source. The resulting wave hits the wall of the vesicle at 45 degrees below the pore and disappears. Exactly the same happens in Wireworld++, where there are also two separate arrangements of weak cells corresponding to two BZ vesicles in which collisions happen.

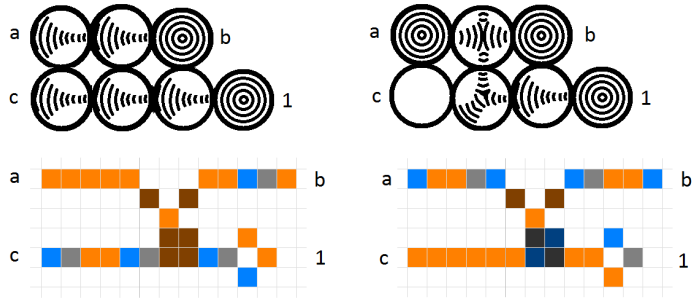

Figure 22. Similarities between implementations of a NAND gate in the BZ reaction [32] and in Wireworld++.

Similar analogies can be considered between other orthogonally arranged uniform BZ gates and Wireworld++ constructs. Therefore, it is possible to use Wireworld++ for simulating computing devices in a sub-excitable, light-sensitive BZ medium.

7. Conclusions and Future Work

In this paper we introduced Wireworld++, a cellular automaton that is more convenient for simulating digital electronic circuits than conventional Wireworld, especially when circuits have intersections of unconnected wires. Most logic gates implemented in Wireworld++ utilize fewer cells than similar gates in Wireworld. We also showed that there is close analogy in the construction and behavior of Wireworld++ gates and corresponding gates implemented by colliding wave fragments in a sub-excitable, light-sensitive Belousov–Zhabotinsky (BZ) medium. Therefore, Wireworld++ can be used for simulating computations in reaction-diffusion systems organized in a square grid of vesicles.

Most cellular automata models of digital computations in reaction-diffusion systems use a hexagonal grid. There are cellular automata for modeling logic gates on a hexagonal grid, a well-known example being the spiral rule cellular automaton [33–37]. We have not seen any applications of cellular automata for modeling BZ gates on a square grid, and we think that Wireworld++ is suitable for this role. Possible future work would be generalization of Wireworld++ to the hexagonal grid, because the design of Wireworld-type automata is different from the spiral rule cellular automaton. It would therefore offer an alternative simulation scheme of these systems. Computing by traveling wave fragments in networks of BZ vesicles with irregular diameters, connection angles and pore efficiencies has also been studied [32]. Generalization of Wireworld++ to such grids is another possible future direction.

In our work, we used the software package Golly [9]. This is very convenient software that allows simulation of a wide variety of two-dimensional cellular automata. However, it is restricted only to square grids. Redesign of this software for arbitrary tessellations would be desired. Feijs worked on simulating analog electronics with cellular automata [16]. His work and ours can be continued for design of an automaton capable of simulating both digital and analog systems, so it could simulate analog-to-digital and digital-to-analog converters and electronics based on them. Finally, both conventional Wireworld and our Wireworld++ are valid only for small currents, because the smallest period of a signal in these cellular automata is $n = 3$. Electronic circuits use thicker wires and proper insulation to

handle large currents. In Wireworld++, using thicker conductors still does not allow signals of period less than 3. Therefore, generalization to larger currents will require creation of a new automaton with its own states and rules.

References

[1] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[2] A. K. Dewdney, "The Cellular Automata Programs That Create Wireworld, Rugworld and Other Diversions," *Scientific American*, **262**(1), 1990 pp. 146–149.

[3] M. Gardner, "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American*, **223**(4), 1970 pp. 120–123.

[4] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems*, Cambridge, UK: Cambridge University Press, 1998.

[5] D. Moore and M. Owen. "The Wireworld Computer." (Jan 26, 2018) www.quinapalus.com/wi-index.html.

[6] N. Malizia. "Simulate Logic Circuits Using Wireworld Cellular Automaton," *Unnikked* (blog). (Jan 26, 2018) unnikked.ga/simulate-logic-circuits-using-wireworld-cellular-automaton-11391bc26b1c.

[7] N. Heise. "3-Tick Logic." (Jan 26, 2018) www.heise.ws/threeticklogic.html.

[8] J. Blandy. "NickGardner.mcl." (Jul 17, 2017) github.com/jimblandy/golly/blob/master/src/Patterns/WireWorld/NickGardner.mcl.

[9] A. Trevorrow, T. Rokicki, T. Hutton, D. Greene, J. Summers, M. Verver, R. Munafo, B. Bostick and C. Rowett. "Golly." (Jan 26, 2018) golly.sourceforge.net.

[10] GitHub. "The WireWorld++ Rule File for the Golly Software." (Mar 3, 2018) github.com/burubaxair/wireworld-pp.

[11] K. Kuratowski, "Sur le problème des courbes gauches en topologie," *Fundamenta Mathematicae* (in French), **15**(1), 1930 pp. 271–283.

[12] E. O. Hwang, *Digital Logic and Microprocessor Design with VHDL*, CL Engineering, 2005.

[13] J. D. Plummer, M. Deal and P. B. Griffin, *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*, Upper Saddle River, NJ: Prentice Hall, 2000.

https://doi.org/10.25088/ComplexSystems.27.1.19

[14] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation* (rev. 2nd ed.), Piscataway, NJ: IEEE Press; Hoboken, NJ: Wiley-Interscience, 2008.

[15] K. Scherer. "WireWorld Gates and Gadgets" from the Wolfram Demonstrations Project—A Wolfram Web Resource. demonstrations.wolfram.com/WireWorldGatesAndGadgets.

[16] L. M. G. Feijs, "Reinventing Electronics with Cellular Automata," *Complex Systems*, **18**(1), 2008 pp. 53–73. www.complex-systems.com/pdf/18-1-3.pdf.

[17] A. Adamatzky, ed., *Advances in Unconventional Computing: Volume 1: Theory*, Switzerland: Springer International Publishing, 2017.

[18] A. Adamatzky, ed., *Advances in Unconventional Computing: Volume 2: Prototypes, Models and Algorithms*, Switzerland: Springer International Publishing, 2017.

[19] E. Katz, ed., *Molecular and Supramolecular Information Processing: From Molecular Switches to Logic Systems*, Weinheim, Germany: Wiley-VCH, 2012.

[20] LifeWiki. "Non-isotropic Life-Like Cellular Automaton." (Jan 26, 2018) conwaylife.com/wiki/Non-isotropic_Life-like_cellular _automata.

[21] M. Walraet. "Wireworld—Un monde cablé." (Jan 29, 2018) matthieu.walraet.net/automate/automate.html.

[22] B. P. Belousov, "Periodically Acting Reaction and Its Mechanism," *Sbornik referatov po radiotsionnoi meditsine*, Moscow: Medgiz, 1958 pp. 145–147.

[23] A. Zhabotinsky, "Periodic Liquid Phase Reactions," *Proceedings of the USSR Academy of Sciences*, **157**(2), 1964 pp. 392–395.

[24] I. Sendiña-Nadal, E. Mihaliuk, J. Wang, V. Pérez-Muñuzuri and K. Showalter, "Wave Propagation in Subexcitable Media with Periodically Modulated Excitability," *Physical Review Letters*, **86**(8), 2001 pp. 1646–1649. doi:10.1103/PhysRevLett.86.1646.

[25] R. Toth, C. Stone, A. Adamatzky, B. de Lacy Costello and L. Bull, "Experimental Validation of Binary Collisions between Wave Fragments in the Photosensitive Belousov–Zhabotinsky Reaction," *Chaos, Solitons & Fractals*, **41**(4), 2009 pp. 1605–1615. doi:10.1016/j.chaos.2008.07.001.

[26] A. Adamatzky and B. de Lacy Costello, "Binary Collisions between Wave-Fragments in a Sub-excitable Belousov–Zhabotinsky Medium," *Chaos, Solitons & Fractals*, **34**(2), 2007 pp. 307–315. doi:10.1016/j.chaos.2006.03.095.

[27] A. Adamatzky and B. de Lacy Costello, "Experimental Logical Gates in a Reaction-Diffusion Medium: The XOR Gate and Beyond," *Physical Review E*, **66**(4), 2002 046112. doi:10.1103/PhysRevE.66.046112.

[28] A. Adamatzky, "Collision-Based Computing in Belousov–Zhabotinsky Medium," *Chaos, Solitons & Fractals*, **21**(5), 2004 pp. 1259–1264. doi:10.1016/j.chaos.2003.12.068.

[29] B. de Lacy Costello and A. Adamatzky, "Experimental Implementation of Collision-Based Gates in Belousov–Zhabotinsky Medium," *Chaos, Solitons & Fractals*, **25**(3), 2005 pp. 535–544. doi:10.1016/j.chaos.2004.11.056.

[30] NEUNEU. "Artificial Wet Neuronal Networks from Compartmentalised Excitable Chemical Media." (Jan 29, 2018) www.neu-n.eu.

[31] A. Adamatzky, J. Holley, L. Bull and B. De Lacy Costello, "On Computing in Fine-Grained Compartmentalised Belousov–Zhabotinsky Medium," *Chaos, Solitons & Fractals*, **44**(10), 2011 pp. 779–790. doi:10.1016/j.chaos.2011.03.010.

[32] J. Holley, A. Adamatzky, L. Bull, B. De Lacy Costello and I. Jahan, "Computational Modalities of Belousov–Zhabotinsky Encapsulated Vesicles," *Nano Communication Networks*, **2**(1), 2011 pp. 50–61. doi:10.1016/j.nancom.2011.02.002.

[33] A. Adamatzky and A. Wuensche, "Computing in Spiral Rule Reaction-Diffusion Hexagonal Cellular Automaton," *Complex Systems*, **16**(4), 2006 pp. 277–297. www.complex-systems.com/pdf/16-4-1.pdf.

[34] A. Wuensche and A. Adamatzky, "On Spiral Glider-Guns in Hexagonal Cellular Automata: Activator-Inhibitor Paradigm," *International Journal of Modern Physics C*, **17**(7), 2006 pp. 1009–1026. doi:10.1142/S012918310600945X.

[35] A. Adamatzky, G. J. Martínez, L. Zhang and A. Wuensche, "Operating Binary Strings Using Gliders and Eaters in Reaction-Diffusion Cellular Automaton," *Mathematical and Computer Modelling*, **52**(1–2), 2010 pp. 177–190. doi:10.1016/j.mcm.2010.02.006.

[36] A. Schumann and A. Adamatzky, "Toward Semantical Model of Reaction-Diffusion Computing," *Kybernetes*, **38**(9), 2009 pp. 1518–1531. doi:10.1108/03684920910991504.

[37] B. de Lacy Costello, R. Toth, C. Stone, A. Adamatzky and L. Bull, "Implementation of Glider Guns in the Light-Sensitive Belousov–Zhabotinsky Medium," *Physical Review E*, **79**(2), 2009 026114. doi:10.1103/PhysRevE.79.026114.