An Assessment of Integrated Digital Cellular Automata Architectures

Victor Zhirnov and Ralph Cavin, Semiconductor Research Corporation
Greg Leeming, Microelectronics Advanced Research Corporation
Kosmas Galatsis, FCRP Center on Functional Engineered Nano Architectonics, UCLA

Future nanoscale technology might drive a migration to different information-processing and computing approaches. One such possibility is the class of digital cellular automata. The recent emergence of multicore architectures, driven by semiconductor technology constraints, motivates the investigation of cellular automata architectures as information-processing alternatives.

With the recent introduction of dual-core processors, the wide-scale transition to parallel processing might have finally begun. How will multicore processing evolve and what will be the dominant computer architecture of the future? As technology reaches the limits of CMOS and beyond, the physical realities of computing hardware might dictate the answer to these questions.

The integration level for nanoscale electronic devices could eventually be in the range of 10^{10} to 10^{11} devices per square centimeter.[1] At this level long interconnects represent a significant challenge to operation (energy consumption), design, and manufacturing (irregular arrays of interconnects with arbitrary connections). Also, nanoscale elements are likely to suffer from significantly higher failure rates than their contemporary counterparts. In addition, low-energy operation requirements and small transistor dimensions are likely to result in higher thermal and quantum error rates. Moreover, the problem of designing complex irregular structures at these density levels is becoming increasingly untenable.

Given these realities, future nanoscale technology may drive a migration to different information-processing and computing approaches. One such possibility is the class of digital cellular automata. The recent emergence of multicore architectures, driven by semiconductor technology constraints, motivates the investigation of cellular automata architectures as information-processing alternatives.

This possible migration to new computing architectures is theoretically predictable. Gianfranco Bilardi and Franco Preparata[2] elegantly argued that, as computing technology approaches the physical limits of scaling (as dictated by the speed of light), ultimate device size reduction limits, and realizable fan-out and fan-in device constraints, it will naturally drive architectures of practical interest toward regular arrays of locally connected computational elements.

CELLULAR AUTOMATA ARCHITECTURES

Cellular automata architectures are possible alternatives to so-called von Neumann architectures. Historically, von Neumann proposed both approaches in the early 1950s.[3]

As Figure 1a shows, the traditional von Neumann or sequential architecture refers to a computer that consists of a processing unit (PU) and memory (M) unit. The memory stores both instructions and data, and the sequence of logical operations is based on flags and control lines. Such a system of *localized* memory and logic units can implement a *general-purpose computer.*

On the other hand, as Figure 1b shows, cellular automata are composed of identical cells with the same connec-
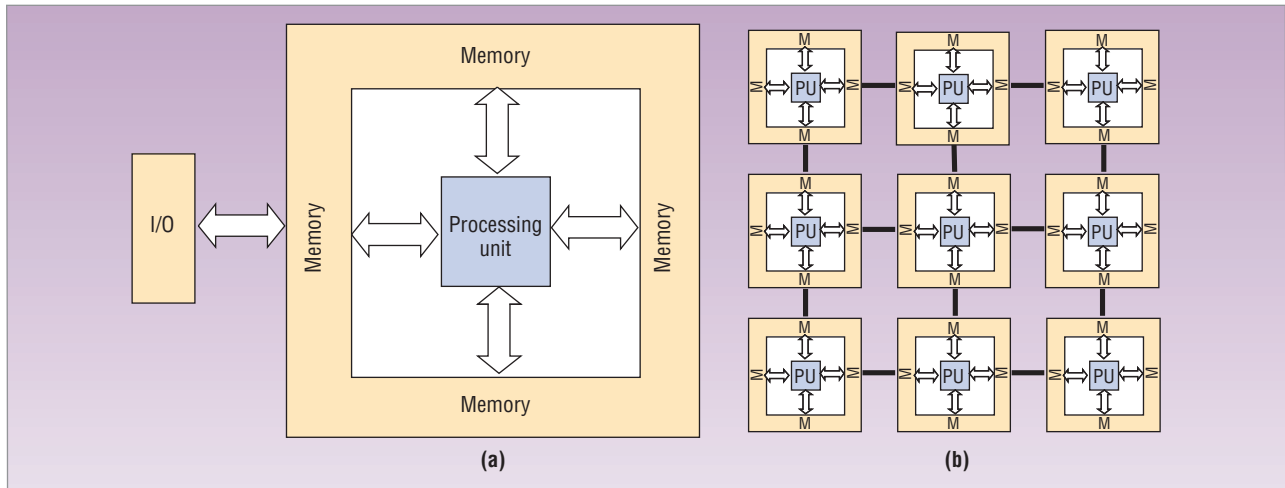
Figure 1. Two generic computer architectures: (a) von Neumann sequential architecture, and (b) cellular array architecture.

tion structure to neighbors. Only a cell's state can distinguish it from its neighbors. Each cell's state is a certain function (a "rule") of the states of neighboring cells.

CELLULAR AUTOMATA FOR MAINSTREAM COMPUTING

In our endeavor to explore whether cellular automata could be a viable mainstream computing implementation, we seek to answer the following key questions:

- What is the minimal required internal complexity of cells in cellular automata that can support general computation?
- How many cells are required for digital cellular automata architectures to become as computationally efficient as von Neumann architectures for general computation?
- Can beyond-CMOS devices provide a substantive computational benefit to cellular automata architectures?
- Is interconnect complexity reduced in a cellular automata architecture?
- Do feasible methods exist to tolerate hard and soft errors in cellular automata?
- Do cellular automata architectures offer power, reliability, design, or manufacturability benefits?

Our analysis will be limited to the case of "digital" cellular automata and is independent of specific hardware implementations.

Minimal complexity cells for cellular automata and cellular arrays

Each cell in a cellular automaton contains a certain number of discrete elements—transistors, resistors, diodes, and so on. The cell's internal complexity—the number of discrete elements—defines the cell function and therefore the operation of the cellular automaton. As von Neumann put it,[3]

… if one constructs the automaton (A) correctly, then any additional requirements about the automaton can be handled by sufficiently elaborated instructions. This is only true if A is sufficiently complicated, if it has reached a certain minimum of complexity.

Essentially, a cellular automata cell must surpass a certain internal complexity threshold, referred to here as "von Neumann's threshold," if it is to perform arbitrarily complex tasks by virtue of elaborate software instructions. For example, let's consider a 1-bit general-purpose processor, which contains an arithmetic logic unit and sufficient memory. We can demonstrate that, with a minimum set of operations, a 1-bit ALU will contain about 98 discrete elements. The addition of essential memory requirements results in a minimum cell complexity on the order of about 150 to 200 elements. This is consistent with von Neumann's estimate, suggesting that the minimum core complexity required to implement general-purpose computing is on the order of a few hundred binary switches.[4]

Cellular automata might not require the full functionality of a general-purpose processor for each cell. In this case, we could implement less-complex cells. In cellular automata, each cell can be in an "alive" or "dead" state, depending on the state of its neighbors.

Consider a two-dimensional array, having eight adjacent cells: N, NE, E, SE, S, SW, W, and NW. The implemented rule must consider $2^8 = 256$ possible combinations of adjacent cell states. Typically one selected rule is applied uniformly to all the cells. We can estimate the minimal cell complexity in a "maximal-rule" cellular automaton as follows:

- The cell senses the independent states of eight neighbors, requiring at least eight elements.
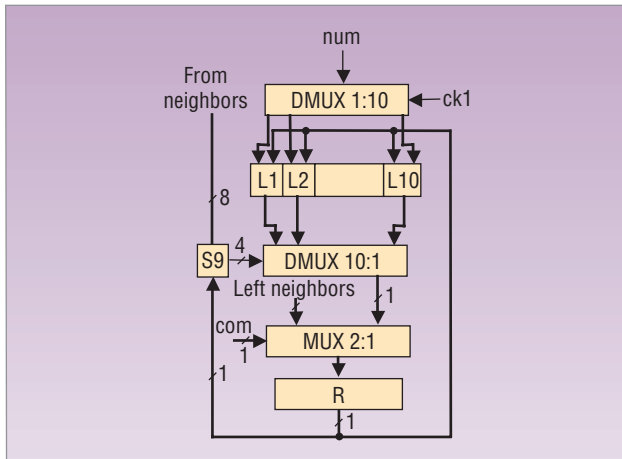- The cell sends information about its own state to its neighbors, requiring at least one element.

Figure 2. Schematic diagram of the cell in a totalistic cellular automaton. (Adapted from M. Dascalu and E. Franti.[5])

- The cell responds to eight independent inputs according to a certain rule using an 8 to 256 decoder, requiring approximately 2,048 elements.
- Cells can be programmed to execute one of 256 rules, requiring at least 256 elements.

We obtain the minimal cell complexity by adding all contributions, and it exceeds 2,300 elements for a 256-rule cellular automaton. This complexity is directly due to the high number of local rules that the cells might need to execute. (Note that the number of elements of these cells is the same as in a basic microprocessor such as Intel's 4004.) High cell complexity and size is acknowledged as the main problem of cellular automata implementations in their generalized form.[5] To reduce the size of cellular automata, the number of rules must be decreased even though this compromises computational efficiency and universality.[5]

For a fixed-rule machine, the 256 discrete elements required for rule programming can be eliminated. Also, one combinational circuit, for example, an AND requiring eight elements at a minimum, can be substituted for the 8 to 256 decoder. For the limiting case of a fixed single-rule cellular automaton with a five-cell neighborhood, the minimal complexity of a fixed-rule cell is greater than $2^4 + 1 = 17$. However, it should be noted that such a machine has limited practical applicability.

One approach is to use a reduced-rule-set cellular automaton implementing the *totalistic* binary functions whose output value depends only on the sum of all of their input variables, not on the value of each input variable.[5] There are several totalistic rules of potential interest, for example, the *majority function*, the *exclusive or*, and the *Game of Life*. Figure 2 shows a schematic diagram of the cell in a totalistic cellular automaton. It consists of a multiplexer 2:1 (MUX 2:1), a demultiplexer 1:10 (DMUX 1:10), a multiplexer 10:1 (MUX 10:1), 10 addressable latches, the S9 circuit (a 9-bits sum), and a 1-bit register.

Hence, we can conclude that a minimum cell will consist of at least several hundred discrete elements, comparable in complexity to the basic single-bit general-purpose computing element.

CELLULAR AUTOMATA IN 2020

A 2006 high-performance MPU contains about 400 million transistors per square centimeter and might approach 7 billion transistors by 2020.[1] If a minimum of approximately 200 components is required per cellular automata cell, future semiconductor processes could conceivably deliver tens of millions of minimal-complexity cells per square centimeter.

How would the performance of a cellular automaton containing this number of cells compare with that of a contemporary 2020 von Neumann architected MPU? Clearly, if the cellular array offers superior performance, we should expect substantive changes in microprocessor design in the future.

We are unaware of an authoritative answer to this question. Most of the literature discussing such comparisons argues that both Turing machines and cellular automata can perform universal computation, where the universal computer has the following capabilities[6]:

- a means of communicating to the outside world with the purpose of receiving input and producing output at any time during computation;
- the ability to perform all elementary arithmetic and logical operations;
- a program made up of basic input, output, arithmetic, and logical operations; and
- an unlimited memory in which programs, the input, intermediate results, and the output can be restored and retrieved.

Several classes of cellular automata have been shown to satisfy the properties of a universal computer, usually by showing equivalence to the Turing machine. The Turing machine has infinite memory by construction whereas universal cellular automata are infinite in extent. Of course, realizable general-purpose computers do not possess this infinite storage capability, yet they can do useful work.

Several specialized algorithms have been demonstrated, for example in fluidics and pattern recognition, where, due to the local nature of the required computation, cellular automata demonstrate significant computational advantage. However, an interesting question is: What hardware complexity does a finite cellular automaton require to obtain equivalence with a simple finite von Neumann computer, for example, a one-bit microprocessor with limited memory? Since it should be fairly straightforward to characterize a one-bit microprocessor, we think this would make an interesting comparison basis for finite cellular automata.

Beyond CMOS nanoelectronic implementations of cellular automata

In digital cellular automata, as in all digital circuits, a binary switch is the fundamental building block. The research community has proposed several alternatives for the CMOS binary switch. All possible realizations of electron-based binary switches face the same limits and tradeoffs for size, speed, and power dissipation.[7,8] The 2005 ITRS reviewed these alternative devices and concluded that none appear to be sufficiently mature to offer competition to the CMOS switch.[1,8]

Interconnects in CA architectures

Figure 3 shows the interconnect length distribution for general-purpose processors.[9] Jeffrey Davis and colleagues offered a rigorous derivation of a complete wire-length distribution for on-chip arbitrary logic networks based on Rent's rule.[9] The theoretical distribution is very close to the empirical relationship shown in Figure 3. In turn, other researchers have shown that a sufficient condition for the appearance of the power-law form of Rent's rule is the statistical homogeneity of gate placement.[10]

We can use the interconnect-length distribution of Figure 3 to calculate an average wire length. In general, the average wire length $L(n)$ is a function of the transistor density, n. Table 1 shows the average wire length $L(n)$ for different n and gate length L_g (calculated using Davis's approach).[9]

We now consider the wire-length distribution in cellular automata architectures. Since we have shown that a cellular automaton cell is likely to have the complexity of a low-end microprocessor, we will assume that each cell requires an interconnect-length distribution equivalent to that of a general-purpose processor and that the interconnects between neighboring cells are minimal—that is, a single conductor. For the four-neighbor case, this results in one interconnect per cell. The assumption of one interconnect between cells is most favorable for cellular automata since this will provide a lower bound on interconnect power dissipation estimates. The intercell interconnect's minimum length is approximately given by the average separation between two neighboring transistors in an integrated circuit. For a statistically homogeneous transistor placement, an average separation between two neighboring transistors L_{t-t} is approximately 10 L_g.

To estimate the average interconnect length in a cellular automata architecture, consider a chip with the total number of transistors N. Let each cell consist of M transistors so that the number of cells is $K = N/M$. If each cell possesses a distribution similar to a general-purpose processor, the interconnect-length distribution
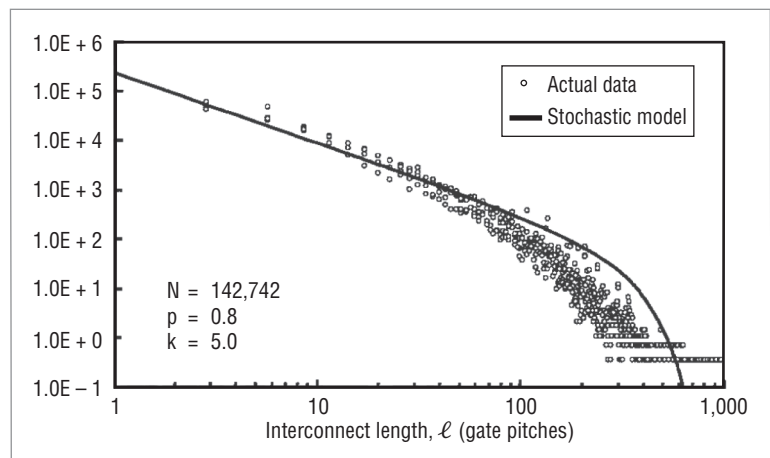

Figure 3. Typical interconnect-length distribution for microprocessors. (Adapted from J.A. Davis, R. Venkatesan, and J.D. Meindl.[9])

inside the cell is typically represented by the distribution shown in Figure 3.

Outside the cells (elementary processors), there are only local short interconnects with length about L_{t-t} (2) between neighboring cells. The number of intercell interconnects is K (one wire per cell). If we consider a maximum-density chip ($N \sim 10^{12}$ in 1 cm^2) organized in a single-core ($K = 1$) and a cellular architecture, the average wire length for a cellular automata architecture is less than that for a single-core architecture, as Table 1 shows.

Whereas the average wire length for a single-core architecture is 4.1, a cellular architecture with a minimum cell complexity, $M_{min} \sim 100$, $(L_1/L_K)_{max}$, is only 2.64. This suggests that, in principle, a cellular implementation can decrease the energy dissipation from interconnects. Note that for larger cell complexity, this ratio naturally would decrease. It should also be noted that additional interconnections might be required for initialization and control lines to program and operate a useful cellular automaton. The presence of such global interconnections might downgrade the potential benefits of cellular automata architectures.

Fault tolerance of cellular automata

A major challenge associated with fine-grained architectures is the foreseeable degree of nanocomponent defects and faults. Several researchers have addressed the fault tolerance of cellular automata.[11-14] Youichi Nishio and Hidenosuki Kobuchi described one early attempt to construct a fault-tolerant cellular automaton.[11] In this model, the maximum number of errors that can be tolerated—that is, corrected—is one in 19 cells. To enable such a level of fault tolerance, each cell needs to be connected to at least 49 neighboring cells—in other words, the very principle of local connectivity is broken. Other researchers have explored fault tolerance in infinite cellular automata[12,14] and have shown that fault-tolerant computation by an infinite cellular automaton is possible in principle.

Table 1. Average wire length in a cellular array chip for different numbers of transistors in a 1 cm² chip *N* and the number of transistors in each cell *M* (the number of cells *K* = *N*/*M*). The average wire length is normalized to the gate length. L_{cell} (*M*) is the average wire length inside the cell, L_K is the average wire length across the chip of *K* cells, and L_l is the average wire length of a single-core architecture (*K* = 1).

N	M	K	L_{cell} (M)/L_g	L_l/L_K
10^{12}	10^{2}	10^{10}	4.1	2.64
10^{12}	10^{3}	10^{9}	5.3	2.09
10^{12}	10^{4}	10^{8}	6.4	1.72
10^{12}	10^{5}	10^{7}	7.5	1.48
10^{12}	10^{6}	10^{6}	8.3	1.33
10^{12}	10^{7}	10^{5}	9.1	1.22
10^{12}	10^{8}	10^{4}	9.7	1.14
10^{12}	10^{9}	10^{3}	10.2	1.09
10^{12}	10^{10}	10^{2}	10.5	1.05
10^{12}	10^{11}	10	10.8	1.02
10^{12}	10^{12}	1	11.1	1.00

However, the results obtained for infinite media might not be relevant for realizable architectures.

Ferdinand Peper and colleagues[14] constructed an asynchronous cellular automaton based on delay-insensitive circuits that was asymptotically fault-tolerant to arbitrary errors in up to one-third of the bits representing the information in the cellular array.

In a good example of the study of robustness in cellular automata that was based on the Game of Life, a classic cellular automaton computational model, the authors studied the time evolution of the model's state at different temperatures.[15] They found that there is a critical temperature at which a given pattern decays and that these decay temperatures are different for different patterns. If the ratio of the "cell local energy"—for example, the energy needed to change the cell state—to the thermal energy is larger than 4.25, all major patterns survive without losing their shape, but for lower ratios, the patterns degrade due to the accumulation of errors. More studies of the time evolution of the patterns in a cellular array at finite temperatures are needed, especially in the application to implementation of specific logic operations.

Power, reliability, design, and manufacturability benefits

The emergence of cellular architectures will also depend on their power consumption, reliability, design complexity, and manufacturability characteristics relative to their von Neumann architecture counterparts. Since cellular architectures have not been developed to the same degree as today's von Neumann architectures,

we can only surmise which features might prove to be advantageous if these architectures do make it into the mainstream.

The thermal performance challenges associated with VLSI design are well known. How will the power requirements and thermal performance of cellular automata compare with von Neumann designs? As suggested, the same devices used to implement von Neumann architectures will likely dominate cellular automata hardware implementations. Therefore, at the component level, cellular automata power requirements and thermal performance are unlikely to be significantly different from von Neumann implementations. However, digital cellular automata might realize a power and thermal advantage at the macro level. The extreme regularity of these architectures should minimize the likelihood of thermal hot spots. Power consumption should be relatively uniform across the chip, resulting in simpler power distribution and heat removal designs.

Reliability is predicted to become increasingly challenging as on-chip components shrink in accordance with Moore's law. Future components are predicted to suffer from higher variability as well as significantly higher failure rates. Cellular automata architectures should make the task of compensating for these future device characteristics easier.

The solution to component variability is likely to be the incorporation of significant compensation circuitry. In a cellular automaton, researchers need to design this specialty circuitry for only a single cell, which will then be duplicated across the entire array. System reliability issues are likely to be addressed through the application of redundancy. The extreme regularity of cellular automata naturally lends itself to redundant designs and should also be advantageous from a manufacturability standpoint.

Researchers have shown that a small cell library of standardized logic blocks (bricks) is sufficient for an efficient implementation of any microchip design.[16] The resultant regular designs lend themselves well to nanoscale manufacturing, relieving the increasing stress on lithography techniques as feature dimensions shrink. Hence, the inherent regularity of cellular automata should be advantageous from a manufacturability standpoint.

We began by seeking to answer six questions about the role of digital cellular automata architectures in future computational architectures, given the rapid advance of VLSI technology. Following in the spirit of the 2006 Focus Center Research Program Workshop on Computation in Nanoscale Dynamical Systems,[17] our approach has been to review the research literature and, where possible, to make esti-

mates of the minimal hardware complexity for minimal cellular automata and von Neumann elements.

We estimate that a flexible-rule cellular automaton would require 2,300 elements for each cell, surprisingly similar to the number of elements in a basic microprocessor such as the Intel 4004. But we could not determine how many cells a digital cellular automata architecture requires to become as computationally efficient as single von Neumann architectures for general computation. We have shown that claims of computational universality for cellular automata are usually based on arrays of infinite extent and have suggested that a cellular automata implementation of a one-bit microprocessor might shed some light on the resolution of this question on general computation.

On the downside, we have not been able to identify non-CMOS technologies that provide a performance or implementation advantage relative to CMOS implementations of cellular automata architectures. For the foreseeable future, we believe cellular automata will be implemented in CMOS technology. Other technologies do not appear to be mature enough to warrant consideration at this time.

On the upside, our analyses indicate that, relative to a single-core von Neumann uniprocessor, cellular automata architectures enjoy a mild advantage in the average length of interconnects. For example, for a minimum cell complexity, $M_{min} \sim 100$, and assuming one billion cells, the cellular architecture implementation's average interconnect length is half that of a uniprocessor architecture. As such, we believe this could translate into energy savings in cellular automata interconnect systems.

While our analysis of tolerance for hard and soft errors in digital cellular automata architectures shows no difference between that in a von Neumann architecture, we believe that they might indeed offer benefits in these areas, primarily due to the innate regularity of the structures. Moreover, assuming uniform cell activity, cellular automata architectures might generate heat more uniformly on the chip and therefore ease heat management challenges.

Semiconductor technology trends favor the realization of regular, locally connected structures, and digital cellular automata conform well to this trend. However, if digital cellular automata are to have an impact on information-processing technology, it is important to demonstrate the capability to address classes of applications of general interest and importance.

In this light, we acknowledge focusing primarily on the hardware issues related to digital cellular automata. When we consider the use of these systems to implement computation for general applications, a vexing set of software challenges arise. For one, a compiler for cellular automata would need to be constructed to set the element rule schedule to implement the prescribed computation, and we are aware of little work in this area. ▪

References

1. Semiconductor Industry Assoc., *International Technology Roadmap for Semiconductors*, *2005 Edition*; www.itrs.net/reports.html.
2. G. Bilardi and F.P. Preparata, "Horizons of Parallel Computation," *J. Parallel and Distributed Computing*, vol. 27, no. 2, 1995, pp. 172-182.
3. J. von Neuman, *Theory of Self-Reproducing Automata*, Univ. of Illinois Press, 1966.
4. J. von Neumann, *The Computer and the Brain*, Yale Univ. Press, 1959.
5. M. Dascalu and E. Franti, "Implementation of Totalistic Cellular Automata," *Proc. CAS 2000*, IEEE Press, 2000, pp. 273-276.
6. S.G. Akl, "The Myth of Universal Computation," *Parallel Numerics*, R. Trobec et al., eds., Part 2, Systems and Simulation, Univ. of Salzburg, Austria, and Jozef Stefan Inst., Slovenia, 2005, pp. 211-236.
7. V.V. Zhirnov et al., "Limits to Binary Logic Switch Scaling—A Gedanken Model," *Proc. IEEE*, vol. 91, 2003, pp. 1934-1939.
8. V.V. Zhirnov et al., "Emerging Research Logic Devices," *IEEE Circuits & Devices Magazine*, vol. 21, 2005, pp. 37-46.
9. J.A. Davis, R. Venkatesan, and J.D. Meindl, "Stochastic Multilevel Interconnect Modeling and Optimization," *Interconnect Technology and Design for Gigascale Integration*, J.A. Davis and J.D. Meindl, eds., Kluwer Academic Publishers, 2003, pp. 219-262.
10. P. Christie and D. Stroobandt, "The Interpretation and Application of Rent's Rule," *IEEE Trans. Very Large-Scale Integration Systems*, vol. 8, 2000, pp. 639-648.
11. H. Nishio and Y. Kobuchi, "Fault-Tolerant Cellular Spaces," *J. Computer and System Sciences*, vol. 11, 1975, pp. 150-170.
12. P. Gacs, G. Kurdyumov, and L. Levin, "One-Dimensional Homogeneous Media Dissolving Finite Islands," *Problems of Information Transmission*, vol. 14, 1978, pp. 92-96.
13. P. Gacs, "Reliable Computation with Cellular Automata," *J. Computer System Science*, vol. 32, 1986, pp. 15-78.
14. F. Peper et al., "Fault-Tolerance in Nanocomputers: A Cellular Array Approach," *IEEE Trans. Nanotechnology*, vol. 3, 2004, pp. 187-201.
15. S. Adachi, F. Peper, and L. Jia, "The Game of Life at Finite Temperature," *Physica D*, vol. 198, 2004, pp. 182-196.
16. V.R.V. Kheterpal et al., "Design Methodology for IC Manufacturability Based on Regular Logic-Bricks," *Proc. 42nd ACM/IEEE Design Automation Conf.* (DAC 2005), IEEE Press, 2005, pp. 353-358.
17. FCRP Workshop on Computation in Nanoscale Dynamical Systems, 19-20 Jan. 2006; www.fena.org/downloads_public/CNDS%20Workshop%20051115.pdf.

Victor Zhirnov, a program manager at the Semiconductor Research Corporation, is responsible for research in the area of emerging nanoelectronic devices. His research interests include properties of materials at nanoscale, fundamental limits of heat removal, and nanoelectronics. Zhirnov received a PhD in physics from the Institute of Physics and Technology, Moscow. He is a member of the editorial team for the chapter on Emerging Research Devices of the International Technology Roadmap for Semiconductors. Contact him at victor.zhirnov@src.org.

Ralph Cavin is chief scientist for the Semiconductor Research Corporation. His research interests are in semiconductor technologies, information-processing architectures, and computer-aided design. He received a PhD in electrical engineering from Auburn University. He is a Life Fellow of the IEEE. Contact him at Ralph.cavin@src.org.

Greg Leeming is an Intel assignee to the Microelectronics Advanced Research Corporation, where he is the program manager for the Focus Center Research Program. His research interests are in novel computer architectures and information processing. Leeming received an MS in electrical engineering from Brown University. He is a member of the IEEE. Contact him at greg.p.leeming@intel.com.

Kosmas Galatsis is the executive director of the FCRP, FENA, and WIN Centers at the University of California, Los Angeles. His research interests include nanodevices, nanopatterning, and nanoarchitectures. Galatsis received a PhD in electrical engineering from the Royal Melbourne Institute of Technology in Australia. He is a member of the IEEE. Contact him at kos@ee.ucla.edu.