Computation and Nanotechnology:

Toward the Fabrication of Complex Hierarchical Structures

Technical Report UT-CS-08-629

Bruce J. MacLennan[*]

Department of Electrical Engineering & Computer Science
University of Tennessee, Knoxville
`www.cs.utk.edu/~mclennan/`

August 22, 2008

Abstract

Enormous progress has been made in recent years in the nanostructuring of materials, and a variety of techniques are available for fabricating bulk materials with a desired nanostructure. However, the higher levels of organization have been neglected, and nanostructured materials are assembled into macroscopic structures using techniques that are not essentially different from those used for conventional materials. We argue that the creation of complex hierarchical systems, with specific structures from the nanoscale up through the macroscale, and especially post-Moore's Law nanocomputers, will require a close alignment of computational and physical processes.

[*] This report is an unedited draft of portions of the Editorial Preface, "Computation and Nanotechnology," for the *International Journal of Nanotechnology and Molecular Computation* **1**, 1 (January 2009).

1 Hierarchical Assembly

Enormous progress has been made in recent years in the nanostructuring of materials, and a variety of techniques are available for fabricating bulk materials with a desired nanostructure. However, the higher levels of organization have been neglected, and nanostructured materials are assembled into macroscopic structures using techniques that are not essentially different from those used for conventional materials. For example, nanostructured materials may be shaped by machining or molding and assembled by conventional manufacturing techniques. Thus we may have self-assembly at the nanoscale and conventional manufacturing at the macroscale, but no systematic fabrication technology applicable to all scales. Is there an alternative?

Fortunately nature provides a suggestive example, for embryological morphogenesis creates highly complex hierarchical systems, with structures ranging from the nanoscale within cells up through multicellular tissues to the level of gross anatomy. As a significant example, we may take the mammalian nervous system. The brain comprises a number of anatomical regions (the lobes), each comprising hundreds of smaller functional regions (e.g., Brodmann's areas, computational maps), which are structured into macrocolumns, which in turn contain minicolumns, each with a half-dozen or so layers. The minicolumns comprise about one hundred neurons with dendritic trees of characteristic shape (and tens of thousands of synapses), all interconnected in specific ways. At the other end of the scale, the brain itself is part of a nervous system, which includes a highly ramified but organized network of nerves. Thus, embryological morphogenesis provides an inspiring example of how self-organized growth, differentiation, and interaction can produce these complex macroscopic structures from microscopic components. Similarly, the mathematical principles of morphogenesis may be applicable to the fabrication of complex hierarchically-structured artificial systems. The physical realization of these mathematical principles is closely connected to computation, which I will consider next.

2 Post-Moore's Law Computing

The reign of Moore's Law is near its end, and so we must consider the implications of a new regime in computing. Whatever the specifics of new technologies, it is apparent that the quest for greater densities and greater speeds will dictate a closer assimilation between computational processes and the physical processes that implement them. This is because greater densities will require us to use fewer physical devices to implement each element of the computational state (e.g., each number), and because achieving greater speeds will preclude us from using sequences of physical processes to implement elementary computational operations (such as addition). That is, the space and time scales of elementary computational processes will have to approach the space and time scales of the physical systems that realize them.

Nanoscale physical processes have different characteristics from the computational processes on which contemporary computer technology is built (binary digital logic). For example, these physical processes are often *continuous* in both time and physical quantity. Even discrete phenomena (such as molecular interactions within a cell) may be best treated as continuous stochastic processes.

Further, natural processes are fundamentally *parallel* and *asynchronous*. In traditional computer design, however, we have arranged physical processes so that they are synchronized and sequential — because we understand such systems better — but that is wasteful, and in the long run it will be more efficient to learn to take advantage of asynchronous parallel processes for computation.

Contemporary computing, including logic design, is often done in a sort of abstract realm in which there are logical dependencies, but spatial arrangement is ignored. At small time and space scales, however, the *spatial organization* of computational elements becomes much more important. For example, the propagation time of signals is always relevant, but may be exploited for information processing. More generally, the spatial organization of computational elements becomes an essential element in information systems design. Also, molecular computation is not a purely abstract process, but we must consider steric factors, diffusion rates, etc., which may be treated as *problems*, but are better viewed as potential computational *resources*.

Nondeterminism is pervasive at the nanoscale. For example, thermal noise is significant in magnitude and unavoidable, defects and faults are inevitable, and quantum phenomena are often relevant. We may view these characteristics as problems — as implied by such terms as "noise," "defect," and "fault" — or we may view them as free sources of useful randomness, variability, etc.

Chemical reactions always have a nonzero probability of going backward, and so molecular computation cannot be assumed to go monotonically forward. Similarly, many other nanocomputational processes get their direction from energy differences, barriers, etc. and have some probability of reversal. Thus models of nanocomputation must take *reversibility* into account. Again, we can treat it as a problem to be solved (an imperfect realization of an idealized unidirectional process), or we can treat it as a free resource to be exploited. For example, simulated annealing and similar optimization algorithms depend on low-priority uphill transitions to escape from local minima (Kirkpatrick, Gelatt & Vecchi, 1983).

Although we have been inclined to think of computation as an abstract process, nanocomputation requires us to recognize that computation is a dissipative or nonequilibrium *physical* process: potentially nonterminating computation cannot continue without a source of matter or energy. Therefore, in nanocomputation, much more so than in traditional computation, the flow of matter and energy becomes a central factor in information system design. The nanoscale presents novel problems for energy supply, but also new opportunities for sources of matter and energy, including electrical, chemical, optical, thermal, and vibrational energy. Similarly, there are novel problems and opportunities for dissipation of waste energy and matter. At the nanoscale, entropy is both a physical and information-theoretic quantity.

Self-assembly is often treated as an equilibrium process (analogous to a terminating computation), but in a more general sense an assembled nanostructure can be a *stationary state* (or dynamic equilibrium) in a nonequilibrium process, as is often the case in natural, especially living, systems. In effect such a system is continually regenerating its stationary state, so that if it is perturbed (e.g., damaged) it will restore the stationary state. Therefore such systems may be self-healing without an explicit self-repair mechanism.

Further, since a stationary state may be a result of the system's interaction with its environment, as well as of its internal dynamics, a change in the environment can cause a shift to a new stationary state. In this way many natural systems adapt automatically to changes in their environments. Therefore we can exploit the physical medium to implement implicitly important characteristics such as healing and adaptation.

Finally, *quantum effects* are inevitable when dealing with small space scales, time intervals, energies, etc. Thus we need a fundamentally quantum-oriented approach to computation, including techniques to make productive use of quantum phenomena, such as tunneling, exchange interactions, entanglement, and superposition (the goal, of course, of quantum computation).

All these characteristics of nanocomputation can be considered problems to be solved, and this is the approach taken in contemporary computer technology, where devices and circuits are designed to operate "correctly" (i.e., to implement binary digital logic) in spite of these physical characteristics. However, arranging physical processes to implement preconceived ideas of computation is costly in density, speed, and power. Therefore we should, as is sometimes said, "respect the medium" and view its characteristics as resources to be exploited rather than as problems to be avoided. For example, thermal noise may be used as a "free" source of randomness, which is useful for many algorithms (e.g., simulated annealing, stochastic resonance: Benzi, Parisi, Sutera & Vulpiani, 1982; Kirkpatrick, Gelatt & Vecchi, 1983). Low precision real numbers may be represented directly by continuous physical quantities, rather than indirectly, as we do now, with a single number being represented by multiple bits, each implemented by operating one or more continuous physical devices (transistors) in saturated mode. Thus post-Moore's Law computing should seek to make productive use of physical properties and processes in the computational medium.

This increased dependence on physical properties might seem to turn computation into a kind of applied physics, but there is still an important role for computational abstractions. We can see this from the history of contemporary computing technology, for the same mathematical abstraction — Boolean logic — has been used as a model of computation since Boole's *Investigation of the Laws of Thought* (1854), through successive generations of implementation technology, from the mechanical logic of Jevon's *logical piano* (1869), through relays, vacuum tubes, discrete transistors, integrated circuits, and several generations of VLSI. This stable theoretical background has permitted a cumulative investment in Boolean logic and circuit design, providing continuity from one technological generation to the next, and saving us from having to reinvent computer design with each new technology. This is possible because Boolean logic is physically realizable, yet sufficiently abstract that it can be realized by a variety of physical systems.

Therefore, in laying the foundation for post-Moore's Law computing we should seek new models of computation that combine physical realism with sufficient abstractness to be implementable in a variety of physical media. Our models of computation need to be close to the underlying physical realization, but not so close that only one realization is possible. Therefore we should adopt as fundamental computational operations those processes that occur in a wide variety of physical systems or that can be fairly directly implemented in them. For example, diffusion is a common physical process, which occurs in a variety of media, from charge carriers diffusing in a semiconductor to molecules diffusing in a fluid, and it has proved useful for information processing and control in natural and artificial systems; therefore it is a good candidate as an operation in post-Moore's Law computing.

Fortunately nature provides many examples of the use of physical processes for information processing, and these can often be abstracted from their specific physical embodiment and realized in other physical systems. Examples include neural network models of computation, excitable media and reaction-diffusion systems used to control spatial organization, molecular regulatory circuits in cells, intracellular DNA/RNA computing, and embryological pattern formation and morphogenesis. Understanding these systems in information processing terms will show how common physical processes may be exploited to more directly realize information-processing functions, and thus show the way to post-Moore's Law computing technologies.

3 Computational Control of Matter

Nanotechnology and computation interact in another important way, but to see it we have to step back and look at computation from a general perspective. Computation uses *physical* processes to realize *abstract* processes. For example, in manual computation the beads on an abacus or the scales of a slide rule are manipulated to realize abstract mathematical operations, such as addition and multiplication. In an analog computer, electrical processes realize a system of differential equations, and in digital computers electrical switches implement binary logic, in order to process numbers, matrices, characters, sequences, trees, and other abstract objects. What distinguishes these physical processes as *computation* is that their purpose is to realize an abstract (mathematical) process, which is in principle *multiply realizable*, that is, realizable by any physical process that has the same abstract structure (MacLennan, 1994, 2004).

There is a tendency to confuse the physical process of computation with the abstract processes it realizes, and to think of computation as an abstract process itself. This tendency is reinforced by the theory of computation, which is based on abstract (mathematical) machine models (such as the Turing machine), whose purpose is generally expressed as the evaluation of mathematical functions. Therefore it is important to recall that computation is a *physical* process during which real matter and energy are controlled. This is easiest to recognize in analog computers, but even in digital computation an abstract process governs the flow of currents, the movement of electrons, etc.

Computation bridges the abstract and the physical (or, as we might say, the formal and the material). Normally we use the physical processes as a means of realizing an abstract purpose, but we can turn the tables and view the abstract process as a means of achieving a physical effect. This is already the case for a computer's output transducers, which are intended to have a physical effect, but we can extend the idea so that the entire computation is designed for the sake of the corresponding physical processes. While the physical processes in an electronic computer might not seem very useful for purposes other than computing, other computing technologies, such as molecular computation, reorganize matter in ways that can be useful for nanotechnology. An example is *algorithmic self-assembly*, which can be implemented with DNA (e.g., Winfree, 1998; Reif, 1999; Rothemund, Papadakis, & Winfree, 2004; Rothemund & Winfree, 2000).

In this context the idea of a general-purpose computation is especially intriguing, since it implies that a wide variety of physical processes could be programmed much like a digital computer is programmed, and provide a systematic methodology for nanostructure synthesis and control (MacLennan, 2003; von Neumann, 1966; Winfree, 1996).

Computation applied to nanotechnology has different standards and tradeoffs from conventional computation. In traditional applications we want the computation to go as fast as possible (and to dissipate as little power as possible, and to store information as densely as possible), so the common goal is to move as little matter and energy as possible in each computational operation. (This progress can be traced from the movement of relay contacts and changes of magnetization in core storage, through vacuum tubes, discrete transistors, and CMOS, to a possible terminus in single-electron transistors.) We have strived to decrease the matter (and energy) of computation as much as possible in order to approximate pure (immaterial) form.

However, if our *purpose* is to move matter and to create nanostructures with specified physical dimensions, then we may want our computations to move more matter rather than less. (Again, output transducers and actuators provide a familiar example from conventional computing, but here we consider the physical characteristics of the entire computation.) Similarly, we will want our computation to proceed at a rate compatible with its intended physical effect.

Molecular computation, especially DNA self-assembly, provides one of the best contemporary examples of the computational control of matter for nanotechnological purposes (e.g., LaBean, Winfree & Reif, 2000; Reif, 1999; Rothemund, 2006; Rothemund, Papadakis, & Winfree, 2004; Rothemund & Winfree, 2000; Seeman, 1999; Winfree, 1998; Yan, Finkelstein, Reif & LaBean, 2003). Therefore IJNMC especially seeks papers reporting progress in molecular computation, but welcomes work on other computational and non-computational approaches to nanotechnology.

4 Embodied Computation

This more intimate relation between information processing and physical processes is characteristic of *embodied computation*, which refers to the synergistic interaction of formal computation and its material embodiment (cf., Hamann & Wörn, 2007; Stepney, 2004, in press). This concept is inspired by *embodied cognition*, an important recent development in cognitive science and robotics (Brooks, 1991; Clark, 1997; Johnson & Rohrer, 2007; Pfeifer & Bongard, 2007; Pfeifer, Lungarella & Iida, 2007), which addresses the critical role that the body and its physical environment play in cognition (in humans and other animals). One of the insights of embodied cognition is that there is much information that the brain does not have to represent because, in effect, the body and its environment represent themselves, and further that there are many information processing tasks that the brain does not have to carry out because they are effectively realized by physical processes in the body and its environment (Dreyfus, 1979). As a consequence, the cognitive load on the brain is decreased enormously. Embodied computation generalizes this approach to all sorts of information processing and control. Thus embodied computation is an attractive strategy for post-Moore's Law computing in that it supports a greater assimilation between computational and physical processes.

In embodied computation, many useful computational processes come “for free” as physical processes. For example, simulated diffusion has proved useful in a number of applications, including path finding, optimization, and constraint satisfaction (e.g., Miller, Roysam, Smith & O’Sullivan, 1991; Steinbeck, Tóth & Showalter, 1995; Ting & Iltis 1994); in effect it is massively parallel breadth-first search. However, simulating diffusion can be expensive on serial or modestly parallel computers, but it is simple to implement physically, and the parallelism comes for free as a consequence of the parallelism of the physical process. Therefore it is not surprising that nature exploits diffusion (of chemicals or the agents themselves) to solve complex information processing and control problems (e.g., Camazine, Deneubourg, Franks, Sneyd, Theraulaz & Bonabeau, 2001). Further, as mentioned previously, noise is unavoidable, especially at the nanoscale. We can view these stochastic processes negatively, as noise corrupting otherwise perfect representations, and which we strive to eliminate or mitigate, or we can “respect the medium” and exploit them as useful sources of randomness that can be applied to information processing (e.g., in stochastic resonance and simulated annealing: Benzi, Parisi, Sutera & Vulpiani, 1982; Kirkpatrick, Gelatt & Vecchi, 1983). Similarly, unavoidable “error” in the realization of idealized computational processes can be turned to our advantage. Again, nature is a useful model; for example, ants follow their trails imperfectly, and there is variability among ants in trail following, which maintains a certain degree of unbiased search and adaptability in their activity (Camazine & al., 2001).

Nature also provides informative examples of how the physical system may be its own representation, which are relevant to the application of computational ideas in nanotechnology. For example, *stigmergy* refers to the process wherein the “project” undertaken by one or more organisms embodies the information required to continue and complete the project (Camazine & al., 2001). The best-known example is wasp nest building (Bonabeau, Dorigo & Theraulaz, 1999). The partially completed nest itself provides the stimuli that guide the individual wasps in the construction process. Therefore there is no need for the wasps to have representations of the completed nest or of the current state of its construction, or to have an internal “program” for nest construction. In this way, relatively simple agents (with modest information processing capacity) can construct complex, functional structures.

The greatest degree of integration between a computation and its realization occurs when the computation is not controlling some separate physical system, but is rather modifying or constructing the physical realization of itself. That is, the computer and the computation co-create each other. So stated, such a process might seem impossible, but it is the basis of embryological morphogenesis, in which embodied computation creates the physical substrate for later embodied computation. Cells signal each other in order to coordinate the creation and differentiation of new cells, which extend the morphogenetic process. Further, in later developmental stages, neural processes create the nervous system, including the brain. (Thus living systems are described as *autopoietic*, or *self-making*: Maturana & Varela, 1980; Mingers, 1994.) Similarly, in some DNA-based algorithmic self-assembly processes, molecular computation creates the physical structure that supports further computation and assembly (e.g., Barish, Rothemund & Winfree, 2005; Cook, Rothemund & Winfree, 2004; Rothemund & Winfree, 2000).

Therefore we may conclude that the creation of complex hierarchical systems, with specific structures from the nanoscale up through the macroscale, and especially post-Moore's Law nanocomputers, will require a close alignment of computational and physical processes.

5 References

Barish, R.D., Rothemund, P.W.K., & Winfree, E. (2005). Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters*, 5, 2586–2592.

Benzi, R., Parisi, G., Sutera, A., & Vulpiani, A. (1982). Stochastic resonance in climatic change. *Tellus*, 34, 10–16.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, UK: Oxford University Press.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47, 139–59.

Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, G., Theraulaz, J., & Bonabeau, E. (2001). *Self-organization in Biological Systems*. New York: Princeton University Press.

Clark, A. (1997). *Being There: Putting Brain, Body, and World Together Again*. Cambridge: MIT Press.

Cook, M., Rothemund, P.W.K, & Winfree, E. (2004). Self-assembled circuit patterns. In J. Chen and J. Reif (Eds.), *DNA Computing 9* (pp. 91–107). Berlin & Heidelberg: Springer-Verlag.

Dreyfus, H. (1979). *What Computers Can't Do: A Critique of Artificial Reason*. New York: Harper & Row.

Hamann, H., & Wörn, H. (2007). Embodied computation. *Parallel Processing Letters*, 17(3), 287–98.

Johnson, M., & Rohrer, T. (2007). We are live creatures: Embodiment, American pragmatism, and the cognitive organism. In J. Zlatev, T. Ziemke, R. Frank & R. Dirven (Eds.), *Body, Language, and Mind*, vol. 1 (pp. 17–54). Berlin: Mouton de Gruyter.

Kirkpatrick, S., Gelatt, C.D., Jr., & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220, 671–80.

LaBean, T.H., Winfree, E., Reif, J.H. (2000). Experimental progress in computational by self-assembly of DNA tilings. In: E. Winfree & D.K. Gifford (Eds.), *DNA-based Computers V* (pp. 123–140). Providence: American Mathematical Society.

MacLennan, B.J. (1994). Continuous computation and the emergence of the discrete. In K.H. Pribram (Ed.), *Rethinking Neural Nets: Quantum Fields and Biological Data* (pp. 199–232). Hillsdale: Lawrence-Erlbaum.

MacLennan, B.J. (2003). Molecular combinatory computing for nanostructure synthesis and control. In *IEEE Nano 2003 (Third IEEE Conference on Nanotechnology)*. Piscataway: IEEE Press.

MacLennan, B.J. (2004). Natural computation and non-Turing models of computation. *Theoretical Computer Science*, 317, 115–145.

Maturana, H., & Varela, F. (1980). *Autopoiesis and Cognition: The Realization of the Living* (R.S. Cohen & M.W. Wartofsky, Eds.), Boston Studies in the Philosophy of Science, 42. Dordecht: D. Reidel Publishing Co.

Miller, M.I., Roysam, B., Smith, K.R., & O'Sullivan, J.A. (1991). Representing and computing regular languages on massively parallel networks. *IEEE Transactions on Neural Networks*, 2, 56–72.

Mingers, J. (1994). *Self-Producing Systems*. New York: Springer.

von Neumann, J. (1966). *Theory of Self-reproducing Automata* (edited & compl., A.W. Burks). Urbana: University of Illinois Press.

Pfeifer, R., & Bongard, J.C. (2007). *How the Body Shapes the Way We Think — A New View of Intelligence*. Cambridge: MIT.

Pfeifer, R., Lungarella, M., & Iida , F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science*, 318, 1088–93.

Reif, J. (1999). Local parallel biomolecular computing. In: H. Rubin & D.H. Wood (Eds.), *DNA-based Computers III* (pp. 217–254). Providence: American Mathematical Society.

Rothemund, P.W.K. (2006). Folding DNA to create nanoscale shapes and patterns. *Nature*, 440, 297–302.

Rothemund, P.W.K., Papadakis, N., & Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12), 2041–53.

Rothemund, P.W.K., & Winfree, E. (2000), The program-size complexity of self- assembled squares. In: S*ymposium on Theory of Computing (STOC)* (pp. 459–468). New York: Association for Computing Machinery.

Seeman, N.C. (1999). DNA engineering and its application to nanotechnology. *Trends in Biotechnology*, 17(11), 437–43.

Steinbeck, O., Tóth, A., & Showalter, K. (1995). Navigating complex labyrinths: Optimal paths from chemical waves. *Science*, 267, 868–71.

Stepney, S. (2004). Journeys in non-classical computation. In T. Hoare & R. Milner (Eds.), *Grand Challenges in Computing Research* (pp. 29–32). Swindon: BCS.

Stepney, S. (in press). The neglected pillar of material computation. *Physica D*.

Ting, P.-Y., & Iltis, R.A. (1994). Diffusion network architectures for implementation of Gibbs samplers with applications to assignment problems. *IEEE Transactions on Neural Networks*, 5, 622–38.

Winfree, E. (1996). On the computational power of DNA annealing and ligation. In: R.J. Lipton & E.B. Baum (Eds.), *DNA-based Computers* (pp. 199–221). Providence: American Mathematical Society.

Winfree, E. (1998). *Algorithmic Self-assembly of DNA*. Unpublished doctoral dissertation, California Institute of Technology, Pasadena.

Yan, H., Park, S.H., Finkelstein, G., Reif, J.H., & LaBean, T.H. (2003). DNA-templated self-assembly of protein arrays and highly conductive nanowires. *Science*, 301, 1882–4.