How Can We Compute with Arrays of Nanostructures?

Michael Biafore
MIT Lab for Computer Science
Email: biafore@mit.edu

3 Aug 1994

Contents

1 Introduction

1.1 Purpose of this document

In part, the goal of the Ultra Program is to extract useful computation from nanometer-scale effects. To accomplish this goal, those of us who are computer scientists must communicate clearly to those of you who are chemists and device physicists precisely what kinds of "computational primitives" you need to obtain from a nanoscale structure before we can contemplate using it as a building block for ultra-dense, ultra-fast computation.

One such computational primitive is widely known—the three-terminal, transistor-like element. In this report, I describe a new computational primitive—the *Few-body Automaton*—which is the building block for a special kind of cellular automaton. Like the three-terminal element, it is powerful enough to form the basis of the most general-purpose computation. Unlike the three-terminal element, which traces its origin to macroscopic switch mechanisms, few-body automata take their inspiration from the very N-body scattering diagrams that can, in principle, be used to describe many nanometer-scale phenomena. My hope in giving them this form is based on the following expectation:

> *Computer scientists can lighten the burden on the device designers who are trying to build useful computational primitives from nanometer-scale effects by finding new ways to compute with computational primitives that already resemble nanometer-scale effects as closely as possible.*

My objective is to explain what this Few-body computational primitive is—but in just enough detail that you can decide for yourself whether or not it is possible to apply it to nanoscale structures that fall within your own domain of expertise.

1.2 Map of this document

I begin by reviewing some well-known scaling arguments that lead one to believe that nanometer-scale devices and cellular-automaton-like architectures are a natural match.

In Section 3, I explain why—despite these scaling arguments—conventional cellular automata are not a viable architecture for very dense arrays of nanostructures, where the nanostructures interact directly with one another, rather than through wires. In Section 4, I show how, starting with the physical form of these interactions between nanostructures, we arrive at the notion of Few-body Automata.

In Sections 4.4 and 4.5, I explain the operation of Few-body automata and establish the fact that they retain the ability to perform general-purpose computation.

The presentation of the Few-body Automaton primitive concludes with Section 5, *How shall we know a promising nanostructure when we see it?* In that section, I give four criteria that determine whether or not the Few-body Automaton approach is likely to be of use for your particular nanometer-scale structure.

Finally, I close with a concrete example that illustrates, step-by-step, how one might go about using a familiar, old, jurassic-scale phenomenon—magnetic bubbles—to make a computer based on the Few-body Automaton.

2 The era of easy down-scaling is ending

The era in which silicon MOSFETs are steadily scaled down will probably end in the next 10-20 years. Combinatorial analyses of how the number of interconnections scale with the number of gates indicate that as circuit density increases, the most natural architectures have the characteristics of a cellular automaton.

In this section we review some scaling arguments first made by R. T. Bate[2, 3, 4, 16], D. K. Ferry[14, 15] and others[7, 21, 35]. These scaling arguments strongly suggest that the most natural architecture for nanometer-scale devices should have only local interconnections, as does a cellular automaton.

Actually, two kinds of limits to the continued downscaling of device dimensions are apparent:

- limits to scaling individual MOS devices, and
- limits imposed by the proliferation of long interconnections.

2.1 Silicon MOS devices probably cannot be scaled much below gate lengths of 0.1μ.

The scaling of individual MOS devices is limited by the range of achievable materials properties. In order to maintain reliability in the face of thermal fluctuations and manufacturing variations, operating voltages must be kept above a certain minimum, $V_{\min}$. Unfortunately, since the range of achievable dielectric constants is limited, as one continues to operate progressively smaller devices at $V_{\min}$, one inevitably produces larger electric fields within the device. As these fields increase, the physical assumptions on which the logical operation of a MOS-type device is based will eventually break down, and the scaled device becomes computationally useless.

Because improved materials (and the ingenuity of designers) play some role, it is impossible to give a precise limit to the scaling of MOS devices. But a number of careful analyses[9, 30, 19, 4] suggest that channel lengths below about 0.1μ would require techniques so extraordinary as to be uneconomical. In an effort to continue miniaturization past this point, the Ultra program seeks to investigate novel, nanometer-scale devices based on qualitatively different physical principles[31, 23].

2.2 As the number of gates increases, current architectures will have too many long interconnections

The second barrier to continued downscaling, the problem of interconnections, is not fundamentally a physical limit, but an architectural one. Combinatorial analyses[10] have shown that current architectures produce a distribution of wire lengths containing so many long wires that their clock speed would ultimately be limited by τ_{RC}, the time required to charge or discharge the longest wires. However, the length of the longest wires is not a reliable metric. If an architecture required only a few long wires, special techniques could alleviate the delays. A better indicator of the limiting clock speed of a circuit is based on the average length $\langle l \rangle$ of its wires.

The average length can be related to a simple architectural property, the so-called Rent exponent[15], which characterizes how the number P of input/output ports required by a circuit scales with the number of gates G. For a wide variety of circuits, P has been found to follow "Rent's rule":

$$P = kG^{\rho} \tag{2.1}$$

where k is a constant and the Rent exponent ρ is typically between 1/2 and 1.

Consider the architectural assumptions that lead to those two extremal cases. If every gate had its own connection to the outside world, then clearly we would be in the $\rho = 1$ limit. At the other extreme, in a two-dimensional array of gates with area $A \propto G$, only nearest neighbors are connected, so input/output ports only occur along the perimeter, and therefore the two-dimensional array has a Rent exponent $\rho = 1/2$. Current architectures for gate arrays exhibit Rent exponents of $\rho \sim$ 0.6-0.7 [33, 26, 8]. Because ρ is an exponent, this excess over the ideal limit $\rho = 1/2$ becomes significant for large enough arrays.

The Rent exponent can be related to the average wire length, which in turn limits the maximum speed of the system. Donath[10, 13] has shown that the average wire length $\langle l \rangle$ depends on the gate count G and the Rent exponent ρ as[1]

$$\langle l \rangle \sim \begin{cases} G^{\rho - \frac{1}{2}} & \text{if } \rho > 1/2 \\ c_0 & \text{if } \rho \leq 1/2 \end{cases} \tag{2.2}$$

where c_0 is a constant. Modeling a wire as a transmission line with resistance R and capacitance C per unit length yields a simple diffusion equation with solutions characterized by a time delay $\tau_{\mathrm{RC}} \sim \langle l \rangle^2$. This time scale then sets the scale for the maximum clock speed.

[1] Actually, for ρ exactly equal to 1/2, Donath finds $\langle l \rangle \sim \log G$, but the distinction is insignificant for our purposes.

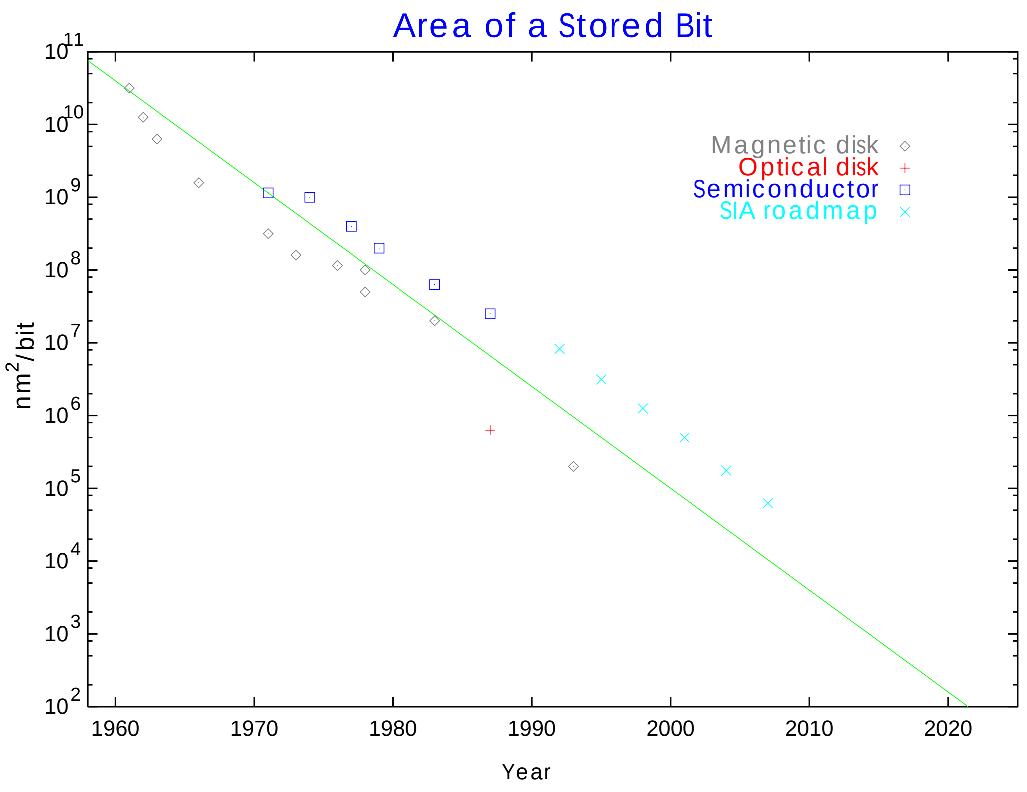

Figure 1: If we are to continue the historical trend of the past trend and the roadmap established by the Semiconductor Industry Association (SIA) for the next 30 years, then we must find physical representations for bits that have characteristic scale 10 nm.

2.3 Scaling analysis points to cellular automata

Letting $\delta \equiv \rho - 1/2$, we see from (2.2) that if $\delta > 0$, then $\tau_{\mathrm{RC}} \sim G^{2\delta}$ eventually diverges as the circuit complexity G grows. As the size G of the circuit increases, the maximum system speed decreases. However, we could prevent interconnection delays from limiting overall system speed if we were able to achieve $p \leq 1/2$. By eq. (2.2), this corresponds to the case of fixed interconnect length. This is the line of reasoning that has led many device physicists to consider modes of computing that require only short local interconnections; that is, cellular automata.

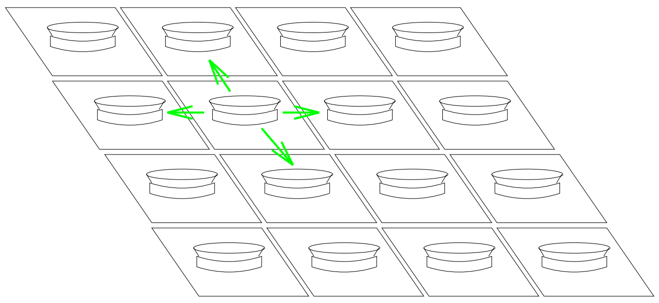

Figure 2: We would like to find a way to arrange nanostructures so that their physical couplings (bold arrows) to nearby cells can be used to implement a general-purpose cellular automaton (cells shown as squares).

3 Cellular automata are computers that bear some resemblance to physical systems...but unfortunately not enough

Despite the scaling arguments of the previous section, conventional cellular automata cannot be used, unmodified, as an architecture for nanometer-scale computation. Their principal defect is that they do not take into account the fact that real physical interactions are not nearest-neighbor-only. As a consequence of this defect, physical implementations of conventional cellular automata do not leave us any way to control *which* nanostructure-cells affect one another, nor do they give us any control we require over *when* the cells affect one another.

3.1 What do we mean by a cellular automaton?

Almost all references to cellular automata are in fact references not to tangible systems, but to abstract models—*mathematical cellular automata* (CA). A mathematical cellular automaton is a discrete dynamical system characterized by

- a discrete space composed of "cells",
- a discrete set of states for the cells, and
- a deterministic "rule" $\widehat{U}$ for updating the state of each cell at discrete time steps.

Mathematical CA resemble physical systems in so far as the cells are uniformly arranged in space and the CA rule—the discrete analog of a physical law—is applied uniformly everywhere.

To distinguish this mathematical abstraction from actual implementations, where nanostructures play the role of the cells, we will call any real or imagined physical implementation of a mathematical cellular automaton a *physical cellular automaton*.

A mathematical CA operates as shown in the space-time diagram of Fig. 3. The state of each cell at time $t_0 + 1$ depends on its own state and the states of its neighbors at time t_0. In a mathematical CA, this dependency is specified by fiat—we simply stipulate that $a_i^{t+1} = f(a_i^t, a_{i+1}^t, a_{i-1}^t)$.

In a physical CA, however, the state of a cell depends on the states of its neighbors according to the physical interactions that couple them. Consequently, it is not within our power to specify an arbitrary dependence. If a physical CA fails to achieve precisely the dependence of the desired mathematical CA, it will fail to compute the same thing as the mathematical CA.

Unfortunately, this is precisely the predicament we face if we naively try to translate a mathematical CA into a physical one. The reason, as first noted by R. Landauer, is simply that

> *Real particles have an interaction that falls off with distance, but is not all that selective, and is not limited to nearest neighbors or next nearest neighbors*[25].

3.2 The problem of screening

At the heart of the predicament lie two problems:

- lack of control over *which* cells affect one another,

and

- lack of control over *when* cells affect one another.

As it turns out, both of these problems can be solved if only we can gain some ability to *screen* the interactions between cells.

The need to control screening is evident from Fig. 4. Suppose we are trying to implement a two-dimensional mathematical CA in which the next state of each cell is supposed to be influenced only by its four nearest neighbors. In the corresponding physical CA (Fig. 4), we will be unable to implement the intended mathematical CA unless we are able to screen the interactions, which almost always persist beyond nearest-neighbors.

Instead, the result will be an irregular, stochastic dynamics with randomly fluctuating neighbors (Fig. 5), a system whose prospects for producing useful computation are considerably dimmer.

Furthermore, the ability to selectively screen interactions between cells would permit us to enforce the third property of CA—discrete time evolution—by screening a cell from even the influence of its nearest neighbors except at discrete times.

In the next section we show one technique for acquiring the ability to screen interactions: We define a new class of cellular automata whose form incorporates Landauer's observation that real interactions do not abruptly vanish beyond the nearest neighbors. The new class of physical cellular automata are called *Few-body Automata*.

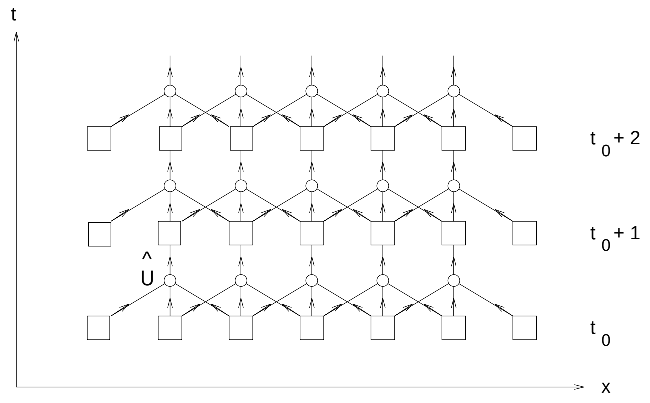

Figure 3: The space-time evolution of a one-dimensional cellular automaton with nearest neighbor interactions.

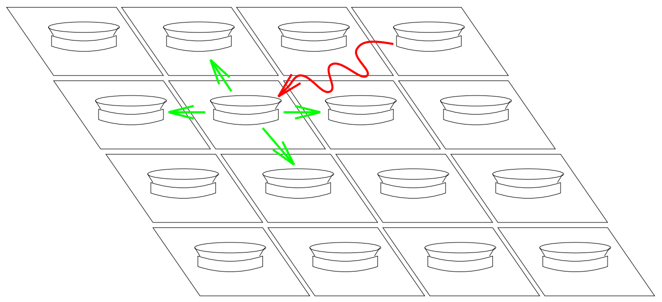

Figure 4: In the absence of effective screening, cells outside the neighborhood of the mathematical CA may, in the corresponding physical CA, have a non-negligible effect (wavy arrow) on the states of cells, thereby corrupting the computation.

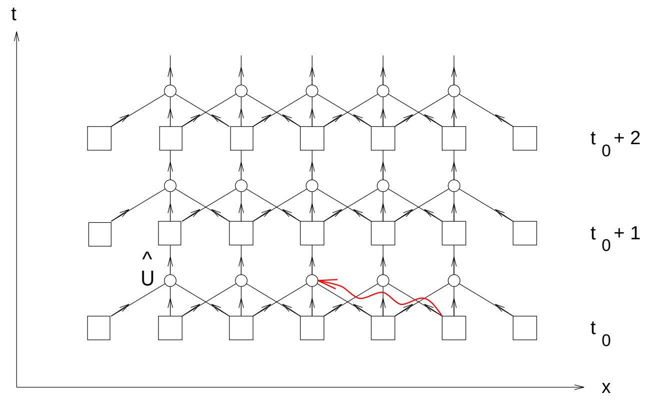

Figure 5: In the mathematical CA, the screening problem manifests itself as an unintended and uncontrollable addition (bold arrow) to the neighborhood of a cell, an addition that, even worse, comes and goes at random times.

4 Few-body automata are computers whose form is derived from the form of physical interactions

We need to replace conventional CA with a new computational primitive that more closely resembles the physical interactions present at nanometer scales. By construction, few-body automata satisfy this requirement.

In this section, we briefly explain how, starting from a very general picture of nanometer-scale interactions, we arrive at the notion of few-body automata.

4.1 The generic form of nanometer-scale computation

In any scheme for performing digital computation, the "bits" of the computation must be encoded in some physical quantity. For example, in MOSFET technology, bits are encoded by collections of charge carriers that raise or lower voltage levels on conductors. To compute at nanometer scales, the bits must be represented by some nanometer-sized entity. The bit-encoding entity might, for example, be a single electron tunneling through a Coulomb-blockade array, a soliton traveling along a polyacetylene molecule, or any localized state passing, bucket-brigade-like through an array of fixed nanostructures.

Computation occurs when the physical quantities representing the bits interact. In a MOSFET, the collection of charge carriers on the gate interacts with the collections of charge carriers on the source and drain by creating a connection between source and drain, a connection created when charge carriers on the gate and in the channel interact via the Coulomb interaction. Generalizing, we obtain a picture (Fig. 6) of the generic nanometer-scale computation as a process in which some localized, nanometer-scale entities affect the information-encoding portion of one anothers state.

4.2 A more restricted, but tractable form

For the most general class of interactions, nothing more specific than Fig. 6 can be deduced about the form of nanometer-scale computation. However, if we restrict ourselves to what are known in scattering theory as *regular* interactions[34], then we can further decompose the shaded blob of Fig. 6 into a composition of several few-body interactions, as in Fig. 7.

This decomposition is permissible only if the interactions among the bit-encoding entities satisfies certain "regularity conditions". Pairwise interactions among N bodies have been shown[12, 22] to be regular if the pairwise interactions satisfy the conditions

$$\lim_{r\to 0} r^{\frac{3}{2}-\epsilon} V_{ij}(r) = 0, \tag{4.1}$$

and

$$\lim_{r\to\infty} r^{3+\epsilon} V_{ij}(r) = 0. \tag{4.2}$$

In essence, these regularity conditions ensure that there is some finite time before and after the interaction beyond which time it is mathematically valid to consider the bit-encoding entities as effectively non-interacting[34, p.26].

Surprisingly, some quite pedestrian interactions fail to satisfy these conditions—for instance, the unscreened Coulomb interaction. And there are some interactions which satisfy them easily, like the screened Coulomb interaction.

4.3 Few-body Automata

Few-body automata can only be defined when the relevant interactions are regular in the sense of Eqs. (4.1–4.2), and henceforth we consider only such interactions. We arrive at the notion of a Few-body Automaton by discretizing the times and places where the interactions of Fig. 7 occur. In one dimension, the resulting pattern of interactions in space-time is shown in Fig. 8. Note that interactions occur in pairs, so this is an example of a 2-body automaton.

Mathematically, the form of few-body automata is the same as that of lattice gases, which have found wide application in the numerical simulation of hydrodynamics[18, 17, 11]. Physically, Fig. 8 represents a process in which pairs of bit-encoding entities—particles—approach one another closely enough that they affect each others state and then diverge from the region where the interaction took place until they approach another particle closely enough to affect its state, whereupon the cycle repeats.

To clarify how Few-body Automata evolve in time, in the next section I describe the physical implementation of a 4-body automaton in two dimensions.

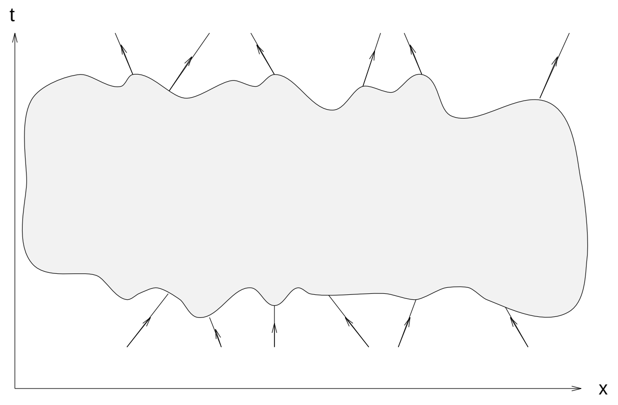

Figure 6: Generic computational process with six input states and six output states.

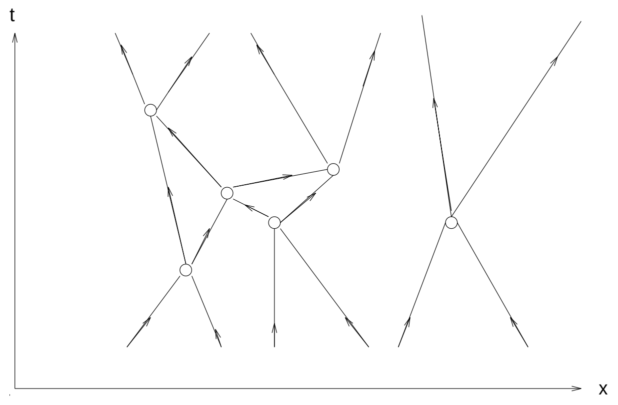

Figure 7: Computational process with six input states and six output states when all interactions are regular.

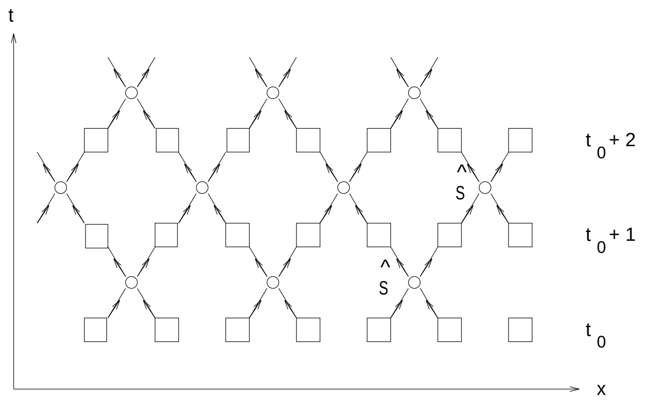

Figure 8: The 2-body automaton that results from discretizing space-time events that involve regular interactions.

4.4 Physical implementation of Few-body Automata

Unlike conventional CA, in which the bit-encoding entities always remain in the cells, in a few-body automaton they must shuttle back and forth between two "clustering locations". In return for this added complexity, however, we gain enough control over which bit-encoding entities interact to extract useful computation from real nanostructures.

In two dimensions, the natural generalization of the one-dimensional 2-body automaton in Fig. 8 is the 4-body automaton of Fig. 9. One possible geometric form that the corresponding physical automaton might take in the xy-plane is shown schematically in Fig. 10. The dynamics is given by a simple two-phase cycle.

4.4.1 Few-body Automata operate in a two-phase cycle

In the first part of the cycle, four bit-encoding entities converge, forming clusters at the vertices labeled S_1. For example, in Fig. 10(a), four bit-encoding entities, with information-bearing states α, β, γ and δ are shown converging to form a cluster at a vertex marked S_1. Elsewhere in the few-body automaton, other groups of bit-encoding entities are simultaneously converging on other vertices marked S_1. Since, by hypothesis, all relevant interactions are regular, the information-bearing states are unchanged until the cluster has formed.

Once the cluster has formed, interactions among the bit-encoding entities change their information-bearing states according to the rule given by the scattering matrix

$$S_1 : (\alpha, \beta, \gamma, \delta) \longrightarrow (\alpha', \beta', \gamma', \delta'). \tag{4.3}$$

After S_1 has "updated" their states, the bit-encoding entities exit the cluster, each heading for one of four different S_2 clusters (Fig. 10(b)). In the S_2 clusters they will participate in a similar process with three new partners. Once the information-bearing states of the bit-encoding entities have again been modifed, they exit the S_2 clusters and again converge on the locations of the S_1 clusters, thus completing the two-phase cycle.

As we will see in Section 4.5, it is possible to make a general-purpose computer even for certain very simple scattering rules S_1 and S_2.

4.4.2 Few-body Automata can avoid the problem of screening

Because the separation D of the clustering locations—unlike the intrinsic range of the physical interactions between bit-encoding entities—is under our control, we have an opportunity to avoid the screening problem of Section 3.2. In particular, we want to choose D large enough that we can neglect all the effects on the information-bearing states of a bit-encoding entity in one cluster that are caused by its interactions with bit-encoding entities in other clusters.

The essential goal in defining few-body automata has been to find a compromise between the physical form of interactions that nature gives and the computational need to maintain some control over when and where the information processing takes place. But before we can be sure we have found a satisfactory compromise, we have to make sure we have not gone so far that we have given up the ability to perform the kind of general-purpose computation. In the next section, we will see that, fortunately, we have not.

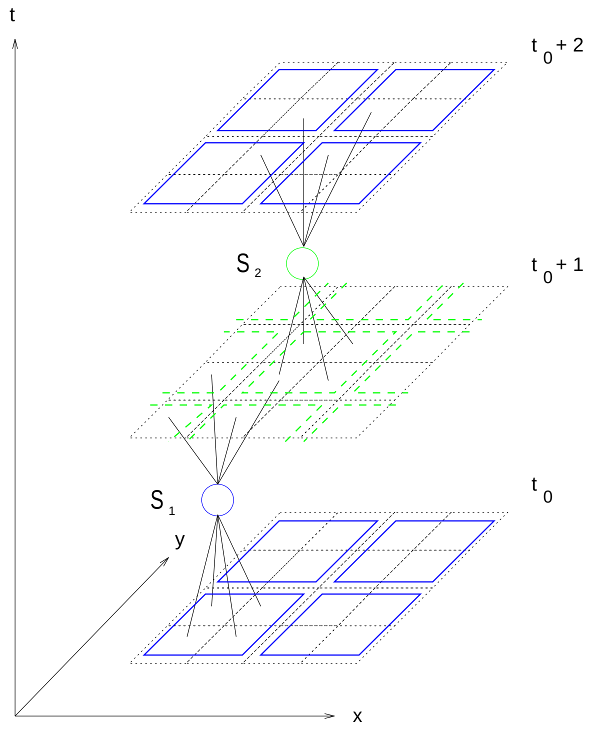

Figure 9: Mathematical few-body automaton with 2×2 cluster neighborhoods. Clusters where S_1 operates are shown as blue squares; clusters where S_2 operates are green; the individual cells of the automaton are shown as dotted boxes.

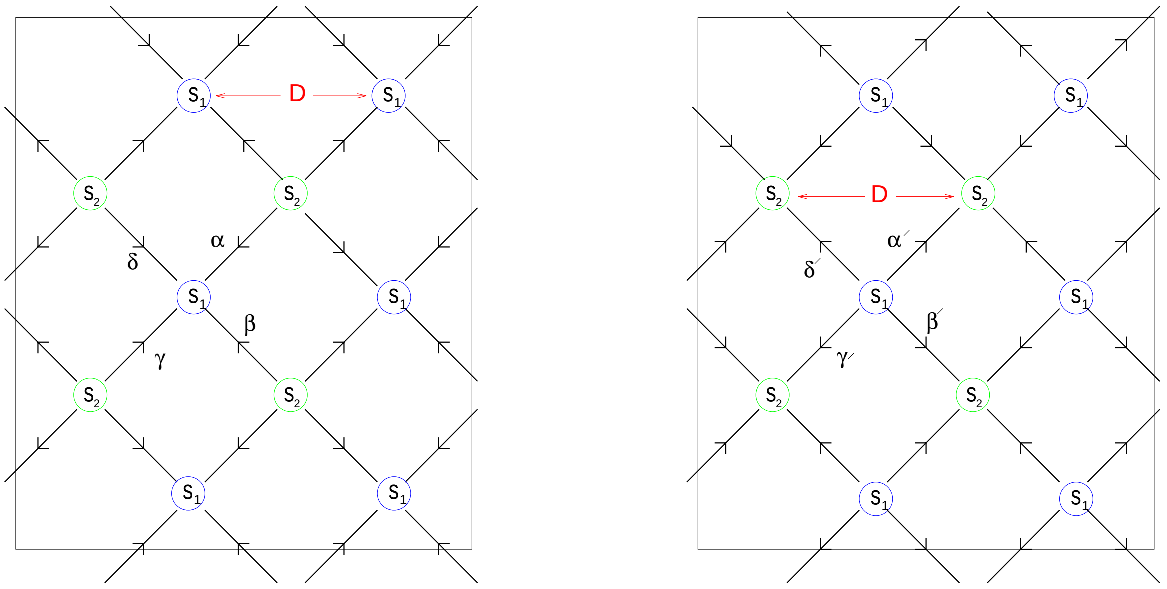

Figure 10: Physical few-body automaton showing flow of the bit-encoding entities at (a) odd time steps, (b)even time steps.

4.5 Few-body Automata can be computation universal

Despite the fact that the definition of Few-body Automata was dictated almost entirely by the form of nanometer-scale interactions, they can efficiently compute the same functions any computer can.

4.5.1 The importance of computation universality

In defining few-body cellular automata in the previous two sections, we have ignored the fact that eventually we want to program these automata to compute. By forcing few-body automata to mimic the form of nanometer-scale interactions, have we inadvertently made it difficult, or even impossible, to use them as a computer? The answer, fortunately is "No".

In theoretical computer science, there is a kind of hallmark for computation. Mechanisms which possess the hallmark can compute anything that any computer can (although it may take a long time to do so), while those that do not have it are demonstrably incapable of performing certain computational tasks. The hallmark is known as *computation universality*.

Almost a decade ago, Margolus[27] showed (by constructing an example) that a kind of automaton called 'partitioning', or 'lattice-gas automata' could attain computation universality. Since one can easily show[5] that the computational properties of Few-body Automata are isomorphic to those of lattice-gas automata, it follows immediately that Few-body Automata can attain the hallmark of universal computation.

4.5.2 The importance of not requiring too many states

We eventually want to use few-body automata as an architecture for turning arrays of nanostructures into computers, and we expect that individual nanostructures will often have only a limited repertoire of accessible states. Consequently, in addition to making sure that Few-body Automata can be computation-universal, we must also make sure that they do not attain their computation universality at the expense of requiring an impossibly large number of states per cell.

For the case of 4-body automata in two dimensions, the "billiard ball" cellular automaton (Margolus[27]), a cellular automaton version of Fredkin's billiard ball model, requires just two states (that is, one bit) per cell, and is computation universal.

The billiard ball rule on the 2×2 clusters is shown in Fig. 11. In the figure, black squares represent a bit value of 1. In this particular case the rule S_1 that operates on the clusters during the odd time steps and the rule S_2 for even time steps are the equal; that is, the four inputs values of a 2×2 cluster are mapped to the same four output values regardless of whether the transformation occurs during an even or odd time step.

If the 1s bits are regarded as the billiard balls and the 0s bits as empty space, this rule inherits its computation-universality from Fredkin's billiard ball model, in which the wires of a digital circuit are represented by streams of infinitely-hard, finite-diameter billiard balls. The presence of a billiard in a particular stream stands for a binary signal value 1, its absence a binary value 0. Where two or more streams cross, a collision will occur if both streams contain a ball and the finite-impact collision will displace the trajectory of both balls. If no collision occurs, the trajectories are unperturbed.

This dependence of the output streams on input streams is called a collision gate. It has been shown that a Fredkin gate can be constructed from collision gates. Since any boolean gate can be constructed from Fredkin gates, the computation-universality of this 4-body automaton follows.

For the case of 2-body automata, I have shown elsewhere[6] that computation universality can be attained in one dimension with fewer than six bits per cell. In two dimensions, 2-body automata can be computation-universal with just two bits per cell.

Consequently, we can rely on Few-body Automata to perform the most general computation possible even when the bit-encoding entities can take on only a few states.

The simple Few-body rule of Fig. 11 therefore constitutes an example of a computational primitive—distinct from the familiar three-terminal behavior—that can serve as an alternative target for designers of nanometer-scale devices.

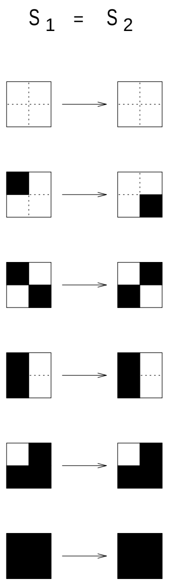

Figure 11: The transition table for the billiard ball CA.

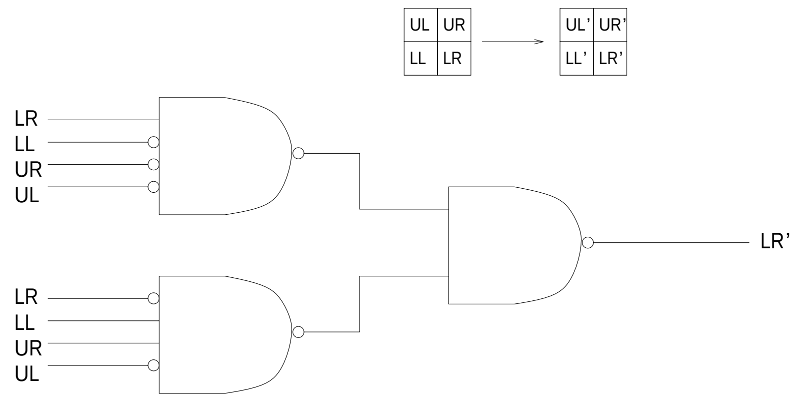

Figure 12: The output of the lower-right cell (LR$\prime$) is a simple Boolean function of the binary input values (LR,LL,UR,UL) of the four cells in the cluster. Here it is written as a composition of AND and NOT functions.

5 How shall we know a promising nanostructure when we see it?

Which nanostructures will be easiest to develop into the Few-body Automaton primitive?

5.1 There must be localized physical states that can represent binary logical states.

The most basic requirement is that we identify some physical states (shown schematically in Fig. 13) that can represent the binary data—the 1s and 0s—of our computation. The entities used to represent binary values should be as small as possible. However, they must also be long-lived, stable states since attempting to encode data in physically unstable states demands elaborate error-correction that can quickly negate any gains in density.

The range of entities suitable for representing binary data is very large. Whereas in larger devices macroscopic currents or voltage levels represent the data, at nanometer scales one might contemplate taking a single electron in a Coulomb blockade array[1, 24] to represent the binary value 1 and the absence of an electron as representing binary value 0. Similarly, others have proposed to encode binary data in solitons propagating along π-conjugated polyenes [7], or as magnetic flux quanta propagating through arrays of coupled Josephson junctions[20].

5.2 It must be possible to transport the localized physical states without changing the part that represents the logical state.

We must be able to do more than just store the data reliably. Even if we aspire "only" to make ultra-dense memories, and not full-fledged computers, the entities representing our data must be able to move without corrupting the data they represent. That is, as shown schematically in Fig. 14, whatever physics governs the dynamics of our bit-carrying entity must permit it to propagate without disturbing that part of the state α that represents the data.

The distance over which the information-bearing states α must propagate freely is determined by the intercell separation D of the few-body automaton (recall Fig. 8).

5.3 The localized physical states must interact via a potential which is not too long-ranged.

The previous two conditions must be met by any nanostructure array before it can be used as a memory. If we further hope to fashion the nanostructure array into a Few-body Automaton, the information-bearing entities must interact with one another via a potential that satisfies the regularity conditions Eqs. (4.1–4.2).

As depicted in Fig. 15 there must be *some* interaction S between the information-bearing entities—shown with incoming states $(\alpha, \beta, \gamma, \delta)$ and outgoing states $(\alpha', \beta', \gamma', \delta')$. The interaction must become strong enough as the information-bearing entities approach one another to affect their information-bearing states. In addition, as the entities part company, the interaction must grow weak rapidly enough that what occurs at one interaction vertex (recall Fig. 10) has a negligible effect on the outcome at a neighboring vertex a distance D away.

5.4 The effect of the interaction on the part of the physical state used to represent the logical state must correspond to a deterministic Few-body rule.

Even after we have decided which physical entities will carry the information, and which of their states will correspond to 0s and 1s, we still have some control over what will occur at the locus of each interaction vertex. Among the "levers" remaining at our disposal are the ability to constrain interactions at each vertex (labeled S_1 or S_2 in Fig. 10) by imposing boundary conditions and external fields.

The final requirement is by far the most challenging. Using only these levers it must be possible to arrange matters in such a way that the net effect of the interaction among information-bearing entities is that a given set of input states always produces the same output states. That is, we must be able to harness the native physical interactions between the information-bearing states in such a way as to synthesize *some* deterministic few-body rule, such as the billiard-ball rule depicted in Fig. 11. Of course, the most desirable rules to synthesize are the computation-universal ones since they permit us to perform general-purpose computation in the nanostructure array.

In the next section, we walk through a detailed example of how these four requirements guide us in designing a computation-universal few-body automaton in a well-known technology—magnetic bubbles.

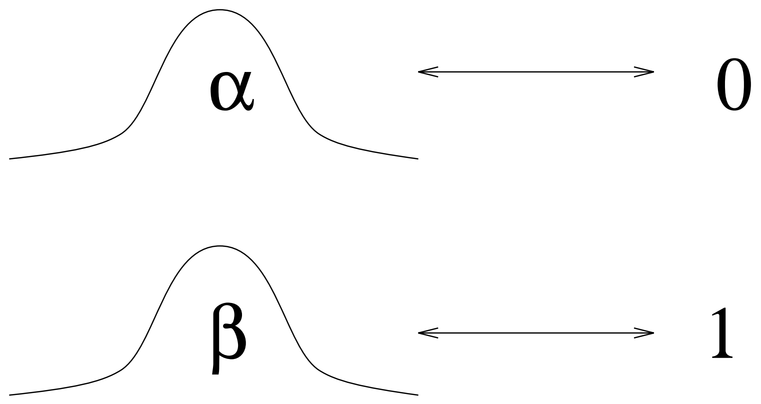

Figure 13: Localized physical states α and β must be associated in some way with binary logical states 0 and 1.

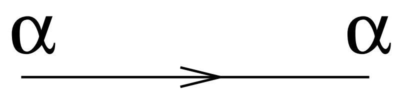

Figure 14: The localized physical states α that are associated with binary values must be able to propagate without changing that value.

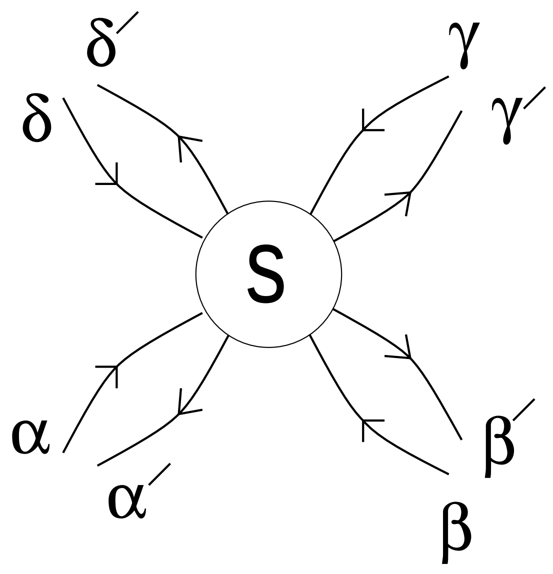

Figure 15: It must be possible to arrange the external fields and boundary conditions in such a way that, when the information-bearing entities approach one another closely, they modify one another's states. For example, the incident states α, β, γ and δ affect one another according to rule S and emerge in states α', β', γ' and δ'.

6 Example: How to make a Few-body Automaton from magnetic bubbles

Magnetic bubbles satisfy the first three criteria in a straightforward manner.

6.1 Bubbles are localized physical states that can represent binary values

As shown in Fig. 16, the binary value 1 is represented by the presence of a bubble domain (bottom of figure), while the absence of a bubble represents binary value 0.

For appropriate values of the anisotropy energy in the plane of the magnetic film, and the static external magnetic field perpendicular to the plane, the bubbles are known to be quite stable.

6.2 Bubbles can be moved without corrupting the binary values they represent

In the absence of time-dependent magnetic fields, the bubbles remain at rest. They can be moved by providing the material in which they reside with an overlay of patterned high-permeability material (e.g. Permalloy) and immersing the bubble film in a rotating in-plane magnetic field. As the field lines up with an elongated part of the permalloy overlay, it concentrates the magnetic field lines, creating a potential gradient and resultant net force on a bubble.

The so-called TI-bar propagation pattern of permalloy overlays in Fig. 17 shows how the information-bearing state, the bubble, can be transported from one point to another so that it arrives in a predictable number of time steps. Since the TI-bar configuration neither spontaneously generate nor destroys bubbles, both the state representing binary value 1 and that representing 0 are faithfully propagated.

Bidirectional propagation (Fig. 18) is achieved by combining a TI-bar pattern with its mirror image. The direction of the rotating in-plane field is shown at lower right, and the direction of bubble propagation along the tracks is indicated by the arrows.

With the bidirectional tracks, we can get the information-bearing entities—the bubbles—into and out of a few-body vertex (shaded region in Fig. 19).

6.3 Bubbles interact via a short-range potential

Two bubbles interact via the magnetic dipole-dipole interaction. Since a bubble domain in the binary 1 state feels a repulsive force in the presence of another domain in the 1 state, but not when the neighboring domain is in the 0 (no bubble) state, the requirement that the information-bearing entities be able to physically sense one another's logical state is also met.

Therefore, the first three criteria given in the previous section are trivially met by magnetic bubbles with the usual identification of a bubble with binary value 1. But what, if anything, can be done to implement a Few-body Automaton rule?

6.4 Can a Few-body rule be implemented?

The last criterion states that we must be able to use the remaining "levers", boundary conditions and external fields, in such a way that the shaded few-body vertex in Fig. 19 causes the bubbles to execute a particular Few-body Automaton rule. In the next section, we observe that the computation-universal rule of Fig. 11 can be implemented using bubble logic circuits found in the literature.

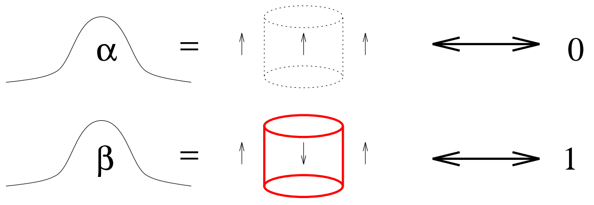

Figure 16: A bubble represents binary value 1, the absence of a bubble value 0.

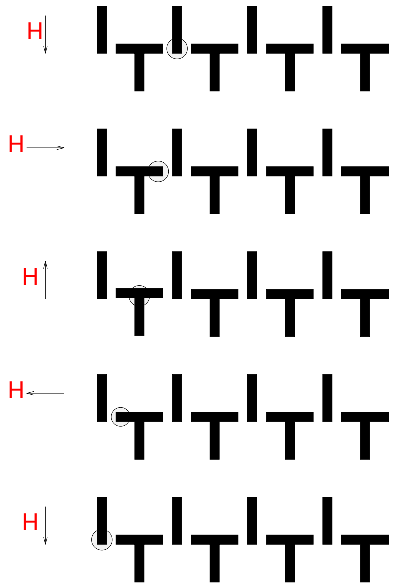

Figure 17: TI-bar track formed by permalloy conductors provides unidirectional propagation of the bubbles without changing their information-bearing state (i.e., their existence).

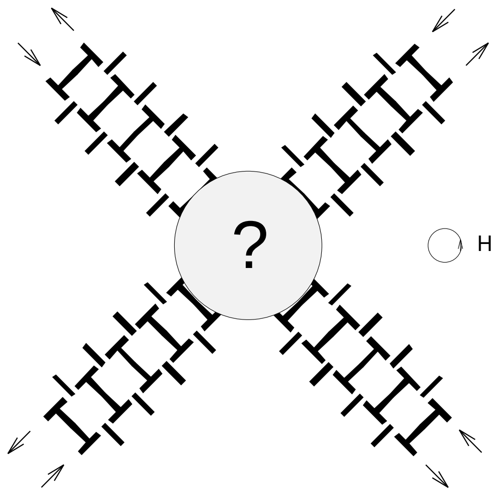

Figure 19: Can we replace the shaded "?" with some pattern of permalloy in such a way that it has the same 4-input/4-output logical function as the universal billiard ball rule of Fig. 11?

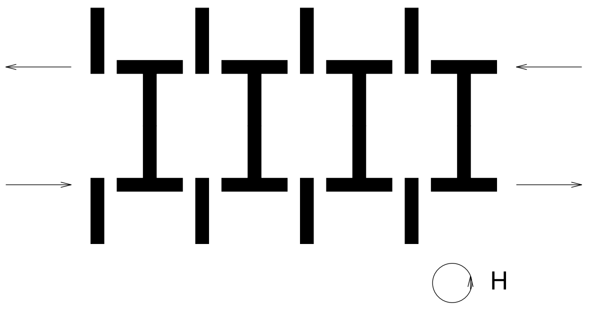

Figure 18: A double TI-bar track provides bidirectional propagation. Arrows show the directions in which bubbles propagate. The in-plane field H rotates counter-clockwise, as in Fig. 17

6.5 Making the Few-body vertex from Bubble Logic

In principle, any logical function can be synthesized from bubble logic, but is it possible to make a *compact* few-body vertex for some computation-universal rule using bubble logic?

Most of the work done on magnetic bubble technology has focused on designing memory, but a few papers on magnetic bubble logic appeared. Perhaps the simplest example of how the dipole-dipole repulsion between bubbles can be harnessed to produce logical functionality is the Bubble Idler shown in Fig. 20. It operates on a pair of bubbles incident sequentially along the left track.

The permalloy pattern is arranged so that an isolated bubble propagates left to right up to the junction, then downward along the vertical track. As the the in-plane field rotates, the first bubble (if present) is captured inside the 4-cycle just below the horizontal TI-bar track. The second bubble is then deflected—that is, continues on the horizontal track—because due to its repulsive interaction with the bubble trapped in the idler. Simultaneously, the repulsive interaction frees the bubble trapped in the idler.

Similarly, Sandfort and Burke[32] have designed an overlay pattern (Fig. 21) that performs both a logical AND and a logical XOR on its inputs. From elementary circuit theory[28], it is known that from these two functions any Boolean function can be constructed.

Therefore, it is evident that, at least in principle, we can find some pattern of permalloy overlays at the vertex that will yield any desired Few-body Automaton rule S (Fig.23). Since, in particular, we can synthesize the computation-universal Few-body rule of Fig. 11, this concludes the construction of a Few-body Automaton in which magnetic bubbles are the bit-encoding entities.

The simplicity of the Few-body rule suggests that, with some ingenuity, a much more compact realization may be possible. If, for example, the vertex can be realized in a region 10nm on a side, then at a clock-rate of 1MHz it should be possible to perform more than 10^{14} bit operations per second on a 1cm$\times$1cm array.

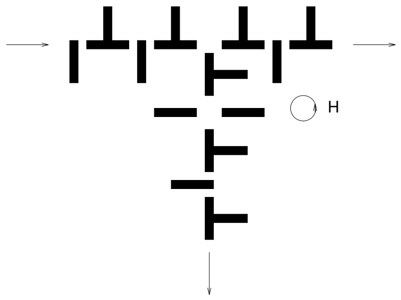

Figure 20: Bubble idler. The first bubble incident from the left is temporarily trapped in the circle of four permalloy conductors (*bottom center*). The second bubble incident from the left is prevented from making the right angle by its repulsive interaction with the trapped bubble. Simultaneously, the trapped bubble is freed, and along the downward track.

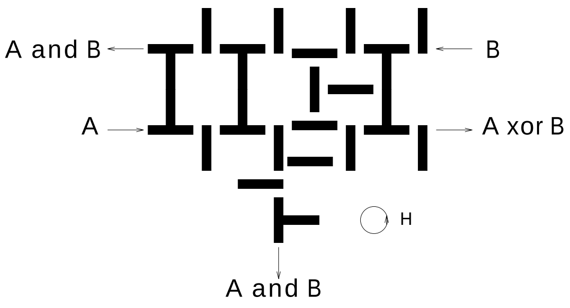

Figure 21: The AND/XOR logic gate of Sandfort and Burke[32].

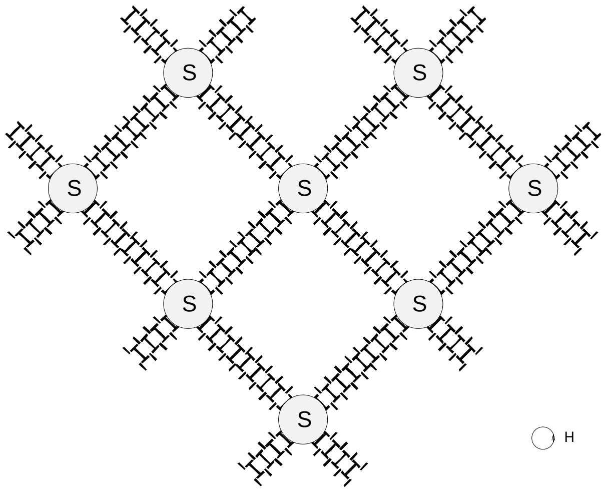

Figure 23: Permalloy circuit of a Few-Body cellular automaton that uses magnetic bubbles.

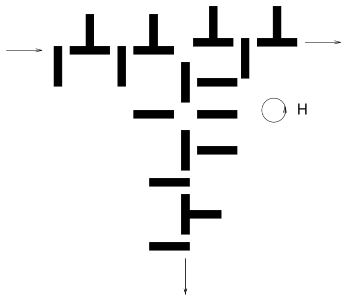

Figure 22: The flip-flop of A. J. Perneski[29].

7 Conclusions

I have described a new computational primitive—the Few-body Automaton—whose form is derived from the form of N-body scattering events. Because their form is derived from the same N-body scattering diagrams used to describe nanometer-scale physics, this computational primitive may be easier to realize in practice for certain nanometer-scale structures.

Like the best-known computational primitive, the three-terminal element, the Few-body Automaton is powerful enough to form the basis of the most general-purpose computation. But it offers designers of nanometer-scale devices the possibility of a different route to extracting useful computational behavior from their structures.

Device designers who want to get some idea of whether or not the Few-body Automaton architecture can be fruitfully applied to their particular nanometer-scale structure should look for three properties:

- The structure must support localized physical states that can represent binary logical states,
- It must be possible to transport the localized physical states without changing the part that represents the logical state, and
- The localized physical states must interact via a potential which is not too long-ranged (as specified by Eqs. 4.1 and 4.2).

Once these three properties are plausibly attainable, the time is ripe for a collaboration between the device designer and CA researchers aimed at overcoming the most significant remaining obstacle to extracting useful computation from arrays of the device, namely that

- The effect of the interactions within isolated clusters of a few such localized physical states must correspond to some computation-universal Few-body Automaton rule.

References

[1] Averin, D. V. and Likharev, K. K., "Single Electronics: A Correlated Transfer of Single Electrons and Cooper Pairs in Systems of Small Tunnel Junctions", In *Mesoscopic Pheomena in Solids* (Altschuler, B. L., Lee, P. A., and Webb, R. A., eds.), 173 (1991).

[2] Bate, R. T., "The future of microstructure technology", *Superlattices and Microstructures* **2**, 9–11 (November 1986).

[3] Bate, R. T., "Nanoelectronics", *Solid State Technology* **32**, 101–108 (1989).

[4] Bate, R. T., Frazier, G. A., Frensley, W. R., Lee, J. W., and Reed, M. A., "Prospects for quantum integrated circuits", In *Quantum Well and Superlattice Physics* (Dohler, Gottfried H., ed.), 26–34 (1987).

[5] Biafore, Michael, *Few-body Automata: A methodology for extracting computation from physical interactions*, PhD thesis, MIT (1993).

[6] Biafore, Michael, "Universal computation in few-body automata", *Complex Systems* **7**, 221–239 (1993).

[7] Carter, F. L., "The Molecular Device Computer: Point of Departure for Large Scale Cellular Automata", *Physica* **10D**, 175–194 (1984).

[8] Chiba, T., IEEE *Trans. on Comput.* **C-27**, 319 (1975).

[9] Dennard, R. H., "Physical Limits to VLSI Technology Using Silicon MOSFETS", *Physica B* **117**, 39–43 (1983).

[10] Donath, W. E., "Placement and Average Interconnection Lengths of Computer Logic", IEEE *Trans. on Circ. and Sys.* **CAS-26**, 272–276 (1979).

[11] Doolen, G. D., *Lattice Gas Methods for PDE's*, North-Holland, Amsterdam (1989).

[12] Faddeev, L. D., *Mathematical Aspects of the Three-body Problem in Quantum Scattering Theory*, Israel Program for Scientific Translations (1965).

[13] Ferry, D. K., "Two-Dimensional Automata in VLSI", In *Submicron Integrated Circuits* (Watts, R. K., ed.), 377–413 (1988).

[14] Ferry, D. K., Akers, L. A., and Greeneich, E. W., *Ultra Large Scale Integrated Microelectronics*, Prentice-Hall (1988).

[15] Ferry, D. K. and Porod, W., "Interconnections and Architecture for Ensembles of Microstructures", *Superlattices and Microstructures* **2**, 41 (1986).

[16] Frazier, G., Interconnections and Architecture for Ensembles of Microstructures, Technical Report 4769644, U.S. Patent, (1988).

[17] Frisch, U., Hasslacher, B., and Pomeau, Yves, "Lattice-Gas Automata for the Navier-Stokes Equation", *Physical Review Letters* **56**, 1505–1508 (1986).

[18] Hardy, J., Pazzis, O. De, and Pomeau, Yves, "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions", *Physical Review A* **13**, 1949–1960 (1976).

[19] Hart, P. A. H., Hof, T. Van'T, and Klaasen, F. M., IEEE *J. Solid-State Circuits* **SC-14**, 343 (1979).

[20] Hayakawa, Hisao, "Josephson computer technology", *Physics Today* **39**, 46–52 (March 1986).

[21] Heilmeier, G. H., Microelectronics: End of the Beginning or Beginning of the End?, In *Proceedings*, 2–5, IEDM, (1984).

[22] Hepp, K., "On the Quantum Mechanical N-body Problem", *Helvetica Physica Acta* **42**, 425–458 (1969).

[23] Kirk, W. P. and Reed, M. A., *Nanostructures and Mesoscopic Systems*, Academic Press (1991).

[24] Kuzmin, L. S., Delsing, P., and Claeson, T., "Single-Electron Charging Effects in One-Dimensional Arrays of Ultrasmall Tunnel Junctions", *Physical Review Letters* **62**, 2539 (1989).

[25] Landauer, R., "Computation and Physics: Wheeler's Meaning Circuit?", *Foundations of Physics* **16**, 551–564 (1986).

[26] Landman, B. S. and Russo, R. L., IEEE *Trans. on Comput.* **C-20**, 1469 (1970).

[27] Margolus, Norman, "Physics-like models of computation", *Physica D* **10**, 81–95 (1984).

[28] Mukhopadhyay, Amar, *Recent Developments in Switching Theory*, Academic Press (1971).

[29] Perneski, A. J., "Propogation of Cylindrical Domains in Orthoferrites", *IEEE Transactions on Magnetics* **MAG-5**, 554–557 (September 1969).

[30] Prince, J. L., *Very Large Scale Integration*, Springer (1980).

[31] Reed, M. A. and Kirk, W. P., *Nanostructure Physics and Fabrication*, Academic Press (1989).

[32] Sandfort, R. M. and Burke, E. R., "Logic Functions for Magnetic Bubble Devices", *IEEE Transactions on Magnetics* **MAG-7**, 358–361 (September 1971).

[33] Striny, K. M., "Assembly Techniques and Packaging of VLSI Devices", In *VLSI Technology* (Sze, S. M., ed.), chapter 13, 26–34 (1988).

[34] Taylor, J. R., *Scattering Theory*, Academic Press (1972).

[35] Weisbuch, C. and Vinter, B., *Quantum Semiconductor Structures: Fundamentals and Applications*, Academic Press (1991).