Reversibility in optimally scalable computer architectures *

Revision: 1.21

Michael Frank, Tom Knight, Norm Margolus

MIT Artificial Intelligence Laboratory, Cambridge, MA 02139, USA

Abstract. An important goal for computer science is to find practical, scalable models of computation that are as efficient as is permitted by the laws of physics. Physics implies fundamental constraints on the efficiency of computations that produce entropy. As a result, it appears that the most efficient possible computers must use reversible primitive operations which produce arbitrarily little entropy. In this paper we show that a 3-D mesh of reversible processor chips, constructible using existing technology, scales better than any physically possible computer based on irreversible primitives.

1 Introduction

In the quest for ever-more-powerful computers, one compelling ultimate goal is to design a physically implementable, scalable computer architecture that is capable of performing any computation with the maximum asymptotic space-time efficiency permitted by the laws of physics.

The recent studies of quantum computing (*e.g.* [13], brief review in [5]) suggest that if it is possible to implement machines that can maintain and manipulate indefinitely large coherent states, then it is this class of machine that will achieve the asympotically highest possible efficiency on some problems, such as factoring. However, since to our knowledge the practicality and scalability of large-scale coherent computers has not yet been established, in this paper we will primarily consider the maximum asymptotic power of physical systems that are *not* able to utilize large-scale coherent superpositions of computational states; that is, "classical" systems. However we expect that many of our conclusions about the optimal classical systems will carry over to the optimal quantum systems as well.

Due to the finiteness of the speed of light, the most efficient architectures appear to be meshes of processors, communicating locally on a regular grid [9, 15, 3, 16]. For maximum physically-scalable connectivity, the mesh should be three-dimensional.

But with conventional technology, 3-D meshes can lead to a problem. If each computational operation produces some minimum amount of entropy (discarded

* This work was supported by DARPA contract DABT63-95-C-0130.

information), and if the density with which information can exist per unit volume is limited (by the technology and/or by fundamental physics), then, as we will show, it follows that the number of operations per second that can be performed in a computer *per unit area of its enclosing surface* is also limited. Thus in conventional technology, expanding a mesh in the third dimension is, after a point, effectively useless for making computations faster.

Moreover, conventional abstract models of computation fail to permit any way to avoid this problem, because the primitive computational operations specified by those models are irreversible—a given computational state resulting from an operation could usually have been reached from more than one possible immediate predecessor. This is important because the laws of physics are themselves reversible at a microscopic level; any microstate can only be reached by a unique path. Therefore, whenever a physical computer attempts to perform one of the logically irreversible operations specified by its abstract model, the discarded information about the prior state is not physically destroyed, but is merely transfered somewhere else, normally to contribute to the thermal entropy of the computer and its surroundings. This relationship between irreversible computational operations and physical entropy was apparently first pointed out by Landauer '61 [10]. A good, though somewhat dated review of the resulting research is provided by Bennett '82 [2].

Thus, direct execution of conventional irreversible models implies a certain minimum entropy be generated with each operation, proportional to the number of bits of computational state that are discarded by the operation. Moreover, current irreversible circuit technology actually inefficiently generates around 10^8 bits of physical entropy per bit of computational state that is discarded!

However, irreversible operations and the accompanying production of entropy are actually not required for any computation, as was shown by Lecerf '63 [11] and independently by Bennett '73 [1]. Unfortunately, it seems that computations that produce no entropy may, at least in some cases, require either more computational space or more time (by an unboundedly large factor) than their counterparts in (nonphysical) models in which unwanted information is permitted to simply disappear [6].

But it is important to remember that in physics, information is *not* permitted to simply disappear, and that physics itself is the reversible machine upon which any irreversible algorithm must ultimately be run. At worst, a computer that is internally reversible can still dispose of unwanted information in a way analogous to the use of fans or coolant flows in ordinary physical computers, namely by simply moving it out of the way, away from where it was generated, and out to an external environment. So ultimately, reversible models need be no less space or time efficient than any physical implementation of any irreversible model, when one keeps in mind the physical resources consumed in either kind of machine by the storage and transport of entropy.

For many problems, the most time efficient algorithms may turn out to be essentially 2-D algorithms that constantly produce entropy which is removed along the third dimension. But for other problems, reversible algorithms that avoid producing entropy in the first place should be able to effectively use all three dimensions, in contrast to any model based on irreversible primitives. We already know of some important classes of problems for which efficient 3-D algorithms can indeed be completely reversible and, in principle, produce no entropy. In particular, there is volumetric simulation of 3-D reversible physical systems; see the examples in [14]. A reversible model of computation can solve such problems more efficiently than any irreversible machine.

This result is not just theoretical. In our group we have developed an electronic technology called Split-Level Charge Recovery Logic (SCRL) [18, 17] that can implement arbitrary reversible operations, and in which the physical entropy generated per operation, though comparable to normal electronics when running at maximum speed, *scales down as the cycle time increases.* The only lower limit, due to leakage currents through nominally "off" transistors, can easily be made exponentially small and insignificant.

With this technology, the third dimension is no longer useless. Making the third dimension thicker, adding more processors per unit area, still requires the speed of each operation to decrease, but now (as we will show) only in proportion to the *square root* of the thickness. Thus the total number of operations per second per unit area can increase without bound. This technology will be able to execute some problems, such as large 3-D physical simulations, unboundedly faster than any technology based on any irreversible model of computation.

Moreover, we propose that our abstract R3M model of computation, which we will specify in more detail later, is ultimately the most efficient physically implementable classical architecture. On problems where the reversibility of the model does not help, then at worst, the R3M, as we will describe it, should be able to directly simulate any physical implementation of any competing model with no asymptotic inefficiencies.

In view of the apparent asymptotic optimality of the R3M model among scalable architectures, we suggest that designers of parallel algorithms should try targeting the R3M model to see if they might gain real-world efficiency from it. Also, we suggest that computer engineers should try building scalable implementations of the R3M architecture using appropriate electronic device technologies that are already beginning to become available.

As for quantum computers, if large-scale coherence can indeed be stably maintained and manipulated at meaningful scales, it is likely that the most efficient computers will turn out to be coherent 3-D meshes of quantum processing elements. Work done now on algorithms for classical reversible 3-D meshes may give us experience that will help later in designing good quantum reversible 3-D algorithms, if and when a workable technology for their implementation becomes feasible.

2 Reversibility, Information, and Entropy

We start by defining reversibility and informally describing some concepts of information and entropy.

The definition of a reversible operation is very simple. An operation is reversible if it can, in principle, be undone. This means that the transformation performed on the accessible degrees of freedom has a functional inverse. That is, the operation is reversible if, for each state that may result from the operation, there is a unique immediate predecessor state that could have yielded that result under the given operation. Put another way, reversibility is simply determinism looking backwards in time.

A computation is reversible insofar as it is composed entirely of reversible primitive operations; a model of computation, or of a system in general, is reversible insofar as it permits only reversible operations to be performed within the model. Physicists like to say of reversible systems that their "state space is incompressible." Reversibility and determinism are both fundamental properties of microscopic physics. Despite the popular view of quantum mechanics as having overthrown determinism, quantum mechanics can be viewed as perfectly deterministic at the level of the unitary wavefunction evolution. The indeterminism that we perceive in "wavefunction collapse" can be interpreted as only a subjective, emergent phenomenon.

As for information, we can adopt an informal concept of information, relating to information theory's definition based on statistical occurences of symbols. In information theory, the information in a message can be estimated by counting the number of symbols in the message, while giving more weight to the symbols which occur less frequently. A message may contain many symbols, but little information if most of the symbols are the same.

In our more informal concept, the information in a state consists of those bits of state that are actually being used to represent something. If bits of state that are not being used to represent anything are always left containing zeroes, and bits that do represent something are equally likely to be zeros or ones, then a crude measure of the number of bits of information in a state is just double the number of 1 bits that are present.

Even in a deterministic system, the amount of this kind of information that is present can increase over time, because new information can simply be deterministically derived from old information. Similarly, the amount of correlated, redundant information that is present can decrease in reversible systems, because correlated pieces of information can be brought together and the redundant information in them can be un-derived; the reversible operation that does this is just the inverse of the deterministic operation that would have produced the derived copy in the first place.

As an example, suppose a state is representing a pair of integers (A, B). Such a state can be transformed by a deterministic operation to a state representing the triple $(A, B, A \times B)$, which contains "more information," but only of a correlated, redundant type. Furthermore, a redundant state of this form can be transformed by a reversible operation back into the state (A, B) which contains "less information" in the sense of less redundancy. But an arbitrary triple (A, B, C), in which there is no redundancy, cannot be made into (A, B) reversibly, just as (A, B) cannot be turned into (A, B, C) deterministically, with

C being arbitrary.

Now, suppose that a particular computation, or an arbitrary physical system, is entirely deterministic and reversible, but that it happens to produce a piece of redundant, derived information in such a way that the inverse operation that would remove the redundancy is for some reason not effectively available. For example, perhaps the redundant information is moving away from the information that it was derived from, and its motion cannot be turned around. Or, perhaps the redundant information was mixed up with some other, unrelated information by some operation whose inverse, though it theoretically exists, isn't physically available in the forwards time direction.

If, through circumstances like these, a system ends up producing some information that is never un-derived, then we call this information "entropy." Disassociated from the information elsewhere with which it is correlated, it can appear random and meaningless.

This definition of entropy is appropriate both for physical systems and in reversible computation systems. Physical and computational entropy are really the same thing; they are interconvertible. Computational entropy can be turned into physical entropy (although very inefficiently, with current technologies) by allowing the energy involved in representing a bit to dissipate. And physical entropy can be turned back into computational entropy: a thermal system can be cooled by allowing it to interact with and randomize a sufficiently sensitive and volatile digital memory that is initially in a low-entropy state (*e.g.*, all zeros).

Entropy in either physical or digital form is annoying because, once it is created, it permanently increases the complexity of a state and takes up space (physical or computational). If the total space available for storing information is limited, or if we cannot move the excess entropy away quickly enough, from some small region of the system where it is being generated outward to another part of the system where there is more room to store it, then eventually the entropy will overflow its acceptable limits and creep into and alter the state of the part of the system that we care about. (For example, by melting its structure.)

We will see that this becomes a fundamental factor limiting the performance of computational systems that produce entropy. But the creation of entropy in computation is not actually necessary. As we will see, there are universal models of computation whose primitive operations are completely reversible at the level of computational state. And moreover, there are workable technologies for implementing these models that can arbitrarily closely approach perfect physical reversibility as well. That is, the physical entropy produced per operation can be made arbitrarily close to zero.

In addition to physical entropy, there is also computational entropy to worry about. Even if a computer uses only reversible circuits, and it executes all instructions reversibly, if its programming model does not actually make the inverse instructions available to the programmer, then there may be no way to execute a program without continuously producing entropy at the computational level. For example, suppose there is an instruction that computes the logical AND of two registers reversibly by placing the result in a previously-empty location.

But if there is no way to undo this operation, then the results of past AND operations will tend to accumulate in memory over time. Since they are never uncomputed, they are a form of entropy.

But with careful design of the set of computational primitives provided to the programmer, it can be possible to perform any desired high-level computation in a way that produces no computational entropy, and instead reabsorbs all intermediate results back into the computation. However, avoiding all entropy production may hurt other measures of performance, so this strategy is not necessarily always useful.

In the following sections we will see exactly how entropy (either in physical or computational form) limits the efficiency of computation, and how avoiding the production of entropy can allow certain computations to be performed arbitrarily faster than they could be performed otherwise.

3 How entropy limits efficiency

In this section we analyze how the production of entropy in a computer limits its asymptotic efficiency.

We start with some considerations from physics. The laws of quantum mechanics imply that a region of space with bounded energy density can contain only a bounded density of (classical) information. Energy density itself may or may not be fundamentally limited.

In any case, there will certainly be technological limits to the achievable density of information storage, long before we reach any fundamental limits. For purposes of storing entropy, very high densities can be achieved by heating dense materials, but the heat capacity of usable materials is limited, as is the maximum temperature at which such materials can be maintained and transported within a functional computing infrastructure.

For stable, digital storage of information, we probably cannot hope to ever do better than, say, by storing bits in the quantized states of individual atoms, at a density of a few bits per atom. And of course, technology that exists today has its own much stricter limits on achievable density of information.

In any case, let ρ denote the highest spatial density of storage of information/entropy, in any form, that is achievable under a technology in question. Example units for ρ: bits per cubic meter.

Now, consider a computation that produces entropy as it proceeds, and consider the convex hull (least-area convex enclosure) bounding the region of space within which that computation is performed. Given the limit ρ on information density, eventually the capacity for information storage within the convex hull will be exceeded, and in order for the computation to proceed, the entropy must pass outwards through the hull. Asymptotically, the rate at which entropy is produced inside the region cannot exceed the rate at which entropy leaves through the convex hull.

What is the fastest rate at which entropy may leave the hull? We have already assumed a maximum entropy density of ρ. We can also assume some maximum entropy removal velocity v, which is, at most, equal to the speed of light c. If entropy crossing the hull is packed at the maximum density and is moving at the maximum velocity, we get an upper bound on the information flux F per unit area of:

$$F = \rho v. \tag{1}$$

As an example: if entropy is encoded (either thermally or digitally) in some material at the atomic scale at a density around 1 bit per cubic Angstrom, and if the material is passing through the surface at a modest 1 centimeter per second, the flux is 10^{24} bits per square centimeter per second. This roughly corresponds to the maximum entropy flux that might be achieved in a stream of coolant moving at that velocity.

This may sound like it is high, but it is actually only about a factor of 30 times higher than what routinely occurs with more passive entropy-removal methods. A chip dumping 100 W of heat from a 1 cm^2 surface (not uncommon today) to a room-temperature thermal reservoir is emitting internal thermal entropy at something like 3.5×10^{22} bits per second per square centimeter.

The flux F, whatever its exact value, asymptotically limits the rate at which entropy can be produced inside a given region, per unit of convex hull surface area.

If each computational operation, on average, produces some nonzero minimal amount of entropy I, then the limit F on the rate of entropy production per unit area leads directly to a limit r on the total rate of computation within a volume, per unit area of the volume's surface.

$$r = F/I. \tag{2}$$

Recalling our second entropy flux example, of $F \approx 3 \times 10^{22}$ bits/sec/cm^2, corresponding to 100 W/cm^2, if $I = 10^8$ bits/operation (typical entropy generated by today's logic gate operations), we have $r = 3 \times 10^{14}$ operations per second per square centimeter. This may sound high, but with, say, a million logic gates per square centimeter in a flat layer of circuitry, and with each gate operating 100 million times per second (both typical numbers today), this means that, with that technology, the maximum amount of circuitry operable at that speed within a region of space is equal to only *a single layer* of circuitry over the region's convex hull! Note that this limit holds independently of the particular spatial arrangement of the circuitry and cooling mechanisms inside the region.

This scaling of computational capability with convex surface area severely limits the asymptotic efficiency of entropy-producing computers on certain classes of problems. For example, consider the problem of simulating a 3-dimensional array of elements, such as voxels (volume pixels) in a volumetric physical simulation, such as of weather, air flow over a wing, or a nuclear blast. Imagine a cube-shaped simulated space, N voxels across in each dimension, in which each voxel must be updated on each step of simulation. Performing all the voxel updates in the entire space for $\Theta(N)$ simulation steps thus requires $\Omega(N^4)$ operations.[2]

[2] This notation is widespread in computer science. Informally speaking, Θ might be

Suppose we want the simulation of this space to be computed within a region of real space whose longest diameter is bounded by $O(N)$. The convex hull of such a region will have a surface area bounded by $O(N^2)$. Thus the highest sustainable rate of entropy-producing computation within the region is $O(N^2)$. Performing $\Omega(N^4)$ operations at this rate will require $\Omega(N^2)$ time, for an average of $\Omega(N)$ time per simulation step.

That is, continuous updating of an $N \times N \times N$ voxel system within a space of width $O(N)$ requires an average of $\Omega(N)$ time *per step* of simulation, in *any* situation in which there is a lower bound independent of N on the entropy produced by each voxel update operation, and in which there is an upper bound independent of N on the maximum entropy flux attainable per unit area. Note that *any* irreversible technology will indeed have some such lower bound on entropy production per voxel update operation, namely, at least as many bits as are irreversibly discarded during the operation.

Note however that if we could perform each voxel update in some constant time and space with absolutely *zero* entropy production, then these simulations could be performed in $O(1)$ time per step, by simply using $\Theta(N^3)$ processing elements to perform the updates in parallel. However, it may be unrealistic to suppose that any technology could ever achieve this. But we will see that in a realistic reversible technology, in which the entropy per operation, though nonzero, can be made as low as desired, some of these 3-D simulations can actually be run in only $O(\sqrt{N})$ time per step—that is, $\Omega(\sqrt{N})$ times faster than if conventional, irreversible technology is used.

First, however, let us examine a slightly different problem. Above we assumed that we wished the diameter of our computer to be restricted to be propotional to the size of the problem. Without any such restriction, and without specifying a need for communication between parts of the computation, we could spread out $\Theta(N^3)$ processing elements as far apart as needed to assure that entropy removal is not the limiting factor, and perform each simulation step in $O(1)$ time. Indeed, this is practical if the "space" being "simulated" is actually only a large search space, such as a search for primes or cryptographic keys... Such applications are not communication-limited, and thus performance on them need never be heat-limited and reversibility cannot help with speed.

However, if we add the requirement that each voxel update depend on the values of the neighboring voxels, as is typical in real physical simulation problems, we find that although spreading out the computation a little bit can allow the asymptotic time per step to be smaller, reversible technologies can asymptotically still beat any irreversible one.

Consider the irreversible case. Even with local interactions only, after simulating the entire space for $\Theta(N)$ steps, each voxel may potentially affect $\Theta(N^3)$ others over $\Theta(N)$ steps, for $\Theta(N^4)$ total voxel-update operations that may de-

---

read as approximately "a quantity proportional to" and Ω as "a quantity at least proportional to." Later we will also use O meaning roughly "a quantity at most proportional to." For the precise formal definition of the Ω, Θ, O order-of-growth notation, see [4], chapter 2.

pend on the original voxel's value. Similarly, each new voxel value that is computed depends on $\Theta(N^4)$ prior computations over the past $\Theta(N)$ simulation steps.

But in real time t, the localized datum which represents a newly computed voxel value can only possibly have been affected by a region of real space of radius tc (where c is the speed of light), which is bounded by a convex surface area of $O(t^2)$. Only a rate $r = O(t^2)$ of entropy-producing operations can be sustained within such a region, for a total number $O(t^3)$ of operations that can possibly affect the datum. Therefore, performing the $O(N^4)$ operations that are required to compute the voxel value resulting from $O(N)$ simultation steps must require at least real time $t = \Omega(N^{4/3})$.

So, sustained simulation of an $N \times N \times N$ 3-D voxel-based system with local interactions requires at least $t = \Omega(N^{1/3})$ average real time per simulation step, in any situation where there is a lower bound on the entropy generated per operation, and an upper bound on entropy flux. Note that this bound, in contrast to the previous $\Omega(N)$ bound, holds true independently of the shape and size of the region within which the simulation occurs. It is also independent of the arrangement of times and places for the individual computations within this region.

The bound can actually be achieved, using a 3-D array of ordinary irreversible processing elements spaced a distance $\Theta(N^{1/3})$ apart from each other, and performing updates every $\Theta(N^{1/3})$ time units. A single column of processing elements contains N elements, each producing entropy at a rate $\Theta(N^{-1/3})$, for a total rate of entropy production within the column of $\Theta(N^{2/3})$. The surface area available for entropy removal at each end of the column has area $\Theta(N^{2/3})$. Therefore, entropy removal does not prevent running the processors at the given rate of $\Theta(N^{1/3})$ time per step.

We will see that our reversible technology can beat this, and achieve $\Theta(N^{1/4})$ time per step.

4 How reversibility can help

Clearly, if we could perform a computation perfectly *isentropically* (without any production of entropy), this would completely change the above analysis, since there would be no issue of entropy removal. The rate of computation that could be sustained within a given region would then be limited only by the packing density and maximum rate of the individual processing elements.

However, it may be physically impossible to achieve true zero-entropy computation while maintaining a constant non-zero rate of processing. Bennett [2] has described a hypothetical clockwork computer that can operate solely via Brownian motion, and produces no entropy. However, it does not run at any non-zero constant speed; when running purely isentropically, it makes a random walk through the computation, which takes $\Omega(n^2)$ expected time to reach the nth computational step.

Thus, the pure Brownian approach does not lead to time-efficient computation. Another approach is what we might call the "ballistic" approach. If the physical entities that carry information around inside a computer move totally frictionlessly, and interact without any loss of energy, then no entropy need be produced. For example, ballistic motion is the basis for the idealized "billiard ball" model of reversible computation proposed by Fredkin and Toffoli [8].

However, we do not know of any way to make a system that operates purely ballistically, at some constant nonzero speed. Indeed, this may not be physically possible.

Nevertheless, using reversible technologies, we do know how to make the entropy production per operation arbtrarily small, by decreasing the speed at which operations are performed, in proportion to the decrease in entropy per operation. In other words, if an operation is performed over a time t, the entropy I produced by the operation scales as

$$I = \Theta(1/t). \tag{3}$$

Note this scaling is not the same as just decreasing the *rate* of entropy production by performing operations more slowly, with the entropy per operation staying the same.

Bennett's Brownian computer can be operated with speed proportional to entropy per operation, if a small driving force is used to keep the machine running forwards steadily. Fredkin and Toffoli '78 [7] proposed an early reversible electronics that had this scaling, based on inductors and capacitors. Merkle has proposed nanomechanical logic mechanisms [12] that exhibit this scaling as well.

Let us now analyze how this inverse dependence of I on t affects the asymptotic efficiency that can be achieved on the problems discussed in the previous section.

For a given technology, let e denote the constant factor in eq. (3), which we may call the "entropy coefficient." It denotes the entropy produced per primitive operation, per inverse of the time over which the operation is performed. Its units are bits of entropy per Hertz.

$$I = e/t. \tag{4}$$

As an example, we estimate that e for SCRL reversible logic gates built with current VLSI technology and operating at around room temperature is around 0.66 bits per Hertz.

Next, let n designate the average number of primitive operations that are being performed simultaneously at any given time, per unit of convex hull surface area. Units for n might be operations per square centimeter. For example, if all circuits that perform primitive operations are active simultaneously, then n is just the number of such circuits per unit surface area. Remember, this is the area of the entire computer's convex hull we are talking about, not the total area of the individual chips... So n becomes larger if we just pile up more layers of circuit boards inside the surface, for example.

Now, if each individual operation is performed over a time t, the rate of computation per unit area is

$$r = n/t. \tag{5}$$

Now, we just substitute equations (4) and (5) back into the flux-limited rate equation, eq. (2) from the previous section, and get:

$$n/t = \frac{F}{e/t}, \tag{6}$$

which we can transform algebraically into

$$\frac{1}{t} = \sqrt{\frac{F}{ne}} \tag{7}$$

and

$$r = \sqrt{\frac{nF}{e}}. \tag{8}$$

This final equation tells us that with reversible technology, the maximum entropy flux F no longer implies a fixed bound on the rate of computation r per unit of surface area, as was the case in eq. (2) when I was constant. Instead, the total rate r of computation per unit surface area can be increased arbitrarily, by increasing n, the number of operations performed simultaneously per unit area. This can be done by just stacking more layers of circuitry over a surface. Also each layer is run more slowly by a factor $1/\sqrt{n}$. The total rate of computation per unit area then increases propotionately to the square root of the number of simultaneous operations per unit area;

$$r = \Theta(\sqrt{n}). \tag{9}$$

Note that r can also be increased separately from n by just reducing the entropy coefficient e. Different reversible technologies differ widely in their e values, but we currently know of no technology-independent, fundamental physical lower bounds on the value of e.

Now, with this equation in mind, to simulate a $N \times N \times N$ array of voxels in a machine of width $O(N)$, we can just let n grow as $\Theta(N)$, by stacking up n layers of circuitry per unit area, and then t, the time per step of simulation, can scale (see eq. (7)) as $\Theta(1/\sqrt{N})$.

An important caveat is that this approach will generally only work if the voxel update function is inherently an invertible function to start with. Otherwise, updating the voxel discards information about its past state. Even if the information is kept in computational form, instead of being dissipated to heat, it is still effectively entropy, and will still limit the rate of computation just as if irreversible hardware were used. Fortunately, the types of systems we are most interested in simulating, namely real physical systems, are always reversible at some level. Systems that behave chaotically, such as in turbulent flows, may be difficult to simulate reversibly at a coarse-grained level. However, we can use massive parallelism to simulate at a more fine-grained level. Systems of interest usually involve bounded energy densities, and thus bounded information per unit volume, in these systems there is no danger of the entropy per volume element needing to increase beyond bound, and so a purely reversible simulation should be able to keep up with the information flow.

In any case, since some 3-D simulations are indeed reversible, the above analysis substantiates our earlier claim that reversible machines can perform some classes of computations asymptotically faster than irreversible machines, because, as we proved in the previous section, irreversible machines require time N per step of simulation to solve this class of problems on a width N machine, whereas the reversible machine only requires time $\sqrt{N}$.

In the other case we considered, where the size of the machine is not specified to be particularly limited but there is interaction between neighboring parts of the simulated space, we can use a 3-D array of $n \times n \times n$ processing elements, $n = \Theta(N)$, in which t, the time per step, and ℓ, the distance between neighboring processing elements, both scale as $\Theta(N^{1/4})$. The entropy I per processor per operation is thus $\Theta(N^{-1/4})$, the entropy production rate per processor per unit time is $\Theta(N^{-1/2})$, the rate for an entire column of n processors is $\Theta(N^{1/2})$, and with an $\Theta(N^{1/2})$ surface area at the end of each column, the entropy generation rate per unit area is $O(1)$. Thus the simulation can be sustained with this $t = \Theta(N^{1/4})$ time per simulation step, faster than the maximum asymptotic rate of $\Theta(N^{1/3})$ that we established in the previous section for irreversible technologies, in which the minimum entropy generation per operation is independent of t.

Unfortunately for our argument, the difference in speeds between reversible and irreversible approaches in this situation is only a factor of $\Theta(N^{1/12})$. Thus, demonstrating a factor of ten increase in the speed of reversible technology relative to irreversible technology in this scenario requires considering a voxel space 10^{12} (a trillion) times wider. Even worse, with current slow switching technology, but assuming dissipationless light-speed communication, the optimal configuration of large reversible or irreversible heat-limited machines has the processors spread apart fairly widely. For example, for a very simple 3-D voxel update rule, and given the parameters of a typical current CMOS circuit technology and 10 $\mathrm{W/cm^2}$ of surface cooling, we estimated the minimum size of a reversible computer that would be faster than any possible irreversible computer (no matter what size) implemented using standard gates in the same technology. It would require an amount of silicon with about the mass of the entire planet Saturn, spread out with 16 meters of lateral spacing between 100-micron wide processing elements, over a disc having the diameter of the entire solar system, with a total power output 60,000 times that of the Sun! (Running at an 18 MHz clock speed, it would perform an astounding 1.8×10^43 voxel update operations per second!)

The lesson to learn from that amusing calculation is that if we are desperate enough for speed that we are willing to spread out our energy-inefficient computers over such a wide area that heat dissipation is no more of a limiting factor than is communications latency, then in that case, the fastest computers are so large that current reversible technologies such as SCRL do not offer any practical speed benefit.

However, in real applications, we will generally want our computers to be as compact as possible, for reasons beyond just speed, such as portability and ease of manufacture. Thus it seems reasonable to levy the restriction that the diameter of the machine should grow linearly in the diameter N of the simulated space.

In this case, as we saw earlier, reversible machines win by a factor of $O(\sqrt{N})$, so demonstrating a factor of 10 increase in the relative benefit from reversibility only requires a factor of 100 increase in scale. In particular, we calculate that with current VLSI technology and with a heat removal limit of 100 W/cm^2, a surface covered with 100 layers of circuitry packed at the maximum planar circuit density could perform about 10 times more operations per second if it used reversible circuits rather than standard circuits. A machine of this scale could feasibly be demonstrated today within the scope of a university research project.

Furthermore, as reversible computing technologies improve in efficiency in the future, the size of the machines where reversibility wins even if the machines are *not* maximally compact will decrease as well. Suppose irreversible computing technology has improved to the point where the absolute minimum of 1 bit (or, in more traditional units, 9.57×10^{-24}J/K $= k \ln 2$) of physical entropy is generated for each bit of computational information that is discarded. *** continue here with examples from current technology & projected future technologies ***

5 The Reversible 3-D Mesh Model

We now describe in detail a model of computation inspired by physical considerations like those taken into account in the analysis above. The goals for this model will be as follows:

1. It should provide guidelines for the architecture of physically-possible computers.
2. It should provide a relatively simple abstraction of computation, which can be a convenient basis for the design of algorithms to be executed on physical implementations of the model.
3. It should facilitate accurate characterization of the efficiency of algorithms executing on physically possible implementations of the model instantiated at arbitrarily large scales. No algorithm analysis within the model should predict better scaling behavior than is physically achievable.
4. It should describe the most powerful classical model of computation that can be physically implemented. That is, it can run any algorithm with as good an asymptotic scaling of time and space requirements as can be achieved for that algorithm on any physically implementable family of non-quantum computers.

The R3M model. An R3M machine consists of a homogeneous, three-dimensional array of units called processing elements (PEs) arranged on a cartesian lattice. Each PE is contained within a cube-shaped region of edge length d. The regions for neighboring PEs meet one another. Each PE contains M bits worth of computational state information, together with fixed machinery for updating this information. The period between updates to the state information is called one "cycle." The length of a cycle is determined by the PE state and can change, but it must always be a nonzero integral multiple of a minimum period p. All the PEs are synchronized, so that each cycle of each PE starts at a time that is an integral multiple of p after some global starting time.

Running through each PE, in each of the X, Y, and Z directions, is a pathway called an "information conduit." Each conduit carries two flows of information, one in each direction. The information in the conduits travels straight at a constant velocity v. The segment of conduit inside each PE has a fixed information capacity ρ. This is not included in the M bits of the PE itself.

Depending on its internal state, each PE may, at the start of any given cycle, exchange information in selected parts of the conduit for information in other parts of its internal state.

In addition to the information conduits, each PE also contains "entropy conduits." Each segment of conduit has a fixed capacity H for entropy, and entropy passes over into the conduit segments of neighboring PEs at a fixed rate. Each cycle, a quantity ef of entropy is added to the PE's segment of conduit, where e is a fixed constant (the "entropy coefficient") and f is the PE's operating frequency, or the inverse of the cycle time. This simulates physical entropy generation due to dissipative losses proportional to the PE's speed of operation.

Whenever the segments of entropy conduit passing through a PE are full, the entropy generated by the PE on each cycle will replace random parts of the PE's internal state, rather than going into the entropy conduit. This simulates temporary effects of overheating.

The PE is allowed to move information from the information conduits to the entropy conduits if desired, replacing the discarded bits with zeroes, but each bit so transfered requires some constant number I of additional bits of entropy to be added to the entropy conduit, representing the inefficiency of current mechanisms for discarding bits of computational state.

At the edge of the array of PE's, all the entropy coming out of the entropy conduits is dissipated into space, and disappears. The information coming out of the information conduits can be considered output. Input may be sent into the information conduits that are going the other way.

The only parameter of the R3M model that is permitted to change as we scale up to larger problems is the number of PE's across the array in each direction.

Let us now consider to what extent the model meets our specified goals.

5.1 Architectural guidelines

The R3M model does indeed directly suggest a physical architecture for its implementation. The PEs can be reversible CPUs, such as we are designing in our research group. The individual PEs can be allowed to control the clock rate at which they are run.

The information conduits may be implemented using ordinary wires running between the network interfaces of neighboring PEs. The entropy conduits may consist of pipes containing a flow of coolant; at the edges of the array, they can feed through radiators to release the accumulated entropy to the environment.

5.2 Model for programming

The spatial organization of the R3M model may not appear at first to be a particularly convenient model to program. However, any desired programming model can be emulated in software by the R3M, without, we assert, any asymptotic loss of efficiency compared to any other possible physical implementation of the model.

So at worst, a programmer can just run existing programming models on top of the R3M to hide whatever details of spatial organization or reversibility he wishes to ignore. If he does so he will pay the same performance penalties that are incurred by direct physical implementations of the traditional architecture.

Additionally, the R3M offers the new possibility that clever programmers and computer scientists can invent new programming models and algorithms that actually perform asymptotically better on large problems than was possible with any of the existing architectures.

We therefore consider the R3M to be a good low-level hardware architecture, because, we assert it does not hide any of the underlying efficiency of physics (except for quantum coherent effects) from its lowest-level programmers. If those low-level programmers wish to hide some of this efficiency from the higher-level programmers in order to provide a more intuitive traditional programming model, they can still do so.

5.3 Accurate scaling

We have ignored very few physical limitations on asymptotic scaling behavior in our characterization of the R3M model. Of course, the effects of gravity are completely ignored.

Also, we ignore the fact that the transport of information and entropy through the conduits is not really free, but itself must generate some small amount of entropy. However, we feel justified in assuming that these losses can be very small compared to the entropy generation that occurs within the processing elements. If no easier mechanisms for nearly-ballistic information transport mechanisms are available, the information and entropy can potentially be encoded in streams of photons moving through optical fibers or evacuated tubes, with extremely low levels of attenuation.

But other than these factors, we feel we have taken all the important physical limitations into account, and that the performance real physical implementation of the R3M can scale exactly as well as would be predicted from this model, up to extremely large scales.

5.4 Optimality

We believe that the R3M is an asymptotically optimal classical model of computation, because it apparently can efficiently simulate any classical computer. However physical space is used inside any computer, with a particular spatial arrangement of components for memory, logic, communication, or heat removal, we can just allocate PEs in corresponding locations of the R3M to perform the same functions, with at most a constant factor overhead due to the fact that each PE only devotes only some constant fraction of its volume, rather than all of its volume, to each of these functions. Even parts of a computer that don't produce any measurable entropy can be simulated by PEs that have their cycle times turned way up so that they are essentially doing nothing.

The only classical ability that the R3M may be lacking is the ability of chemical systems to move a packet of information (embodied in a molecule, for example) randomly around a space, with no generation of entropy (other than in the position of the molecule) until the packet encounters another piece of information with which it can interact.

It is currently unclear how this sort of behavior can be simulated by anything other than a chemical system in an entropically-efficient fashion. However, it is also unclear whether this means of interaction actually allows any asymptotically more efficient computations than could be achieved by algorithms that bring information together in a more controlled way.

As another caveat, our claim of asymptotic optimality says nothing about the size of the constant factor inefficiencies in a given reversible implementation compared to alternative implementations of non-R3M models. Even though chemical systems may turn out to be no faster asymptotically than an electronic R3M, they may be a significant constant factor faster than an R3M on the problems to which they are most suited, at least until electronic technology approaches the molecular scale as well.

In conclusion of this section, we feel that the R3M model proposed above, although still fairly abstract and sketchy, meets all the goals set forth at the start of this section and is therefore a good candidate for an optimally scalable classical model of computation.

6 Conclusion

In this paper we have presented a strong argument, based on fundamental physical constraints, that at least for large instances of certain classes of (non-quantum) computations, a computer based on reversible primitive operations is faster than any computer that is not.

Reversible computers that demonstrate this performance improvement can actually be implemented by using the SCRL circuit techniques of Younis and Knight [18, 17] together with commercially available VLSI fabrication processes. Large 3-D arrays of reversible processors of this type (perhaps several meters across) can perform physical simulations faster than any conventional computer of comparable size.

Based on these insights, we have proposed a general-pupose architecture and abstract model of computation called the Reversible 3-D Mesh. The model apparently permits expressing any physically possible scheme for the storage, manipulation and transport of both information and the entropy that is inevitably produced whenever information is discarded. We conjecture that therefore the model can solve any class of problems using asymptotically no more time or space than any other non-quantum architecture.

6.1 Future Work

Much work along these lines of course remains to be done. The specification of the operation of individual processing elements in the R3M was intentionally left vague. Just about any universal reversible processor would suffice, but some may be much more convient to program, or more efficient by a constant factor, than others. The best compromise is currently unclear. But a detailed, if tentative, PE specification might be a good starting point for the exploration of these issues.

One very big area for future study is to try optimizing parallel algorithms for various problems to run on the R3M model. Due to the nature of the R3M, this is somewhat like designing specialized hardware to solve the problem, in that one must think about the physical location of information and how it flows through the system. The R3M model makes sure that one does not forget about the constraints imposed by the speed of light and the laws of thermodynamics.

One important advance would be a high-level programming language for expressing these "physical" algorithms. Such a language would do for the design of physical algorithms what current programming languages do for the design of ordinary serial algorithms. The language would provide ways of expressing new high-level abstractions of different ways of organizing the physical flow of information.

And of course, work is needed on implementation technologies. The "entropy coefficient" of current reversible technologies is much too high. However, even with current technology it could be a useful exercise to actually build a large R3M so that people could experiment with programming it.

Finally, the classical approach of this paper begs the question, what about quantum computers? The research on quantum computers is extremely interesting, and if implementation technologies get to the point where the decoherence times are long enough to permit error correction techniques to be applied, we expect that in the long run the asymptotically fastest machines will be quantum computers.

But locality and entropy are realities of quantum mechanics as well as of classical physics, so arguments analogous to those in this paper should still apply, and so if quantum computation works out, we expect that the optimal architecture will likely be a *quantum* reversible 3-D mesh, that directly generalizes our current R3M model.

References

1. C. H. Bennett. Logical reversibility of computation. *IBM J. Research and Development*, 6:525–532, 1973.
2. C. H. Bennett. The thermodynamics of computation, a review. *Int'l J. Theoretical Physics*, 21(12):905–940, 1982.

3. Gianfranco Bilardi and Franco Preparata. Horizons of parallel computation. Technical Report CS-93-20, Brown University, May 1993. http://www.cs.brown.edu/publications/techreports/reports/CS-93-20.html.
4. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. MIT Press/McGraw Hill, 1990.
5. Michael P. Frank. Quantum computation primitives. Area exam paper, http://www.ai.mit.edu/~mpf/qcpaper.html, February 1996.
6. Michael P. Frank and M. Josephine Ammer. Separations of reversible and irreversible space-time complexity classes. Extended abstract to be submitted to CCC-98. http://www.ai.mit.edu/~mpf/rc/memos/M06_oracle.html, 1997.
7. E. F. Fredkin and T. Toffoli. Design principles for achieving high-performance submicron digital technologies. DARPA Proposal, November 1978.
8. E. F. Fredkin and T. Toffoli. Conservative logic. *Int'l J. Theoretical Physics*, 21(3/4):219–253, 1982.
9. W. Daniel Hillis. New computer architectures and their relationship to physics or why computer science is no good. *Int'l J. Theoretical Physics*, 21(3/4):255–262, 1982.
10. R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Research and Development*, 5:183–191, 1961.
11. Y. Lecerf. Machines de Turing réversibles. Insolubilité récursive en $n \in N$ de l'équation $u = \theta^n$, où θ est un "isomorphisme de codes". *Comptes Rendus hebdomadaires des seances de l'academie des sciences*, 257:2597–2600, 1963.
12. Ralph C. Merkle. Two types of mechanical reversible logic. *Nanotechnology*, 4:114–131, 1993.
13. Peter W. Shor. Algorithms for quantum computation: Discrete log and factoring. In *Foundations of Computer Science, Proc. 35th Ann. Symp.*, pages 124–134. IEEE Computer Society Press, 1994.
14. Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT Press, 1987.
15. Paul M. B. Vitányi. Locality, communication and interconnect length in multicomputers. *SIAM J. Computing*, 17:659–672, 1988.
16. Paul M. B. Vitányi. Physics and the new computation. In *Mathematical Foundations of Computer Science, Proc. 20th Int'l Symp. (MFCS)*, volume 969 of *Lecture Notes in Computer Science*, pages 106–128. Springer-Verlag, 1995. http://www.-cwi.nl/~paulv/physics.html.
17. S. G. Younis. *Asymptotically Zero Energy Computing Using Split-Level Charge Recovery Logic*. PhD thesis, MIT Artificial Intelligence Laboratory, 1994.
18. S. G. Younis and T. F. Knight, Jr. Asymptotically zero energy split-level charge recovery logic. In *International Workshop on Low Power Design*, pages 177–182, 1994.