Will Moore's Law be Sufficient?

Erik P. DeBenedictis, Sandia National Laboratories

***Abstract*—It seems well understood that supercomputer simulation is an enabler for scientific discoveries, weapons, and other activities of value to society. It also seems widely believed that Moore's Law will make progressively more powerful supercomputers over time and thus enable more of these contributions. This paper seeks to add detail to these arguments, revealing them to be generally correct but not a smooth and effortless progression.**

This paper will review some key problems that can be solved with supercomputer simulation, showing that more powerful supercomputers will be useful up to a very high yet finite limit of around 10^{21} FLOPS (1 Zettaflops . The review will also show the basic nature of these extreme problems.

This paper will review work by others showing that the theoretical maximum supercomputer power is very high indeed, but will explain how a straightforward extrapolation of Moore's Law will lead to technological maturity in a few decades. The power of a supercomputer at the maturity of Moore's Law will be very high by today's standards at 10^{16}-10^{19} FLOPS (100 Petaflops to 10 Exaflops, depending on architecture , but distinctly below the level required for the most ambitious applications.

Having established that Moore's Law will not be that last word in supercomputing, this paper will explore the nearer term issue of what a supercomputer will look like at maturity of Moore's Law. Our approach will quantify the maximum performance as permitted by the laws of physics for extension of current technology and then find a design that approaches this limit closely.

We study a "multi-architecture" for supercomputers that combines a microprocessor with other "advanced" concepts and find it can reach the limits as well. This approach should be quite viable in the future because the microprocessor would provide compatibility with existing codes and programming styles while the "advanced" features would provide a boost to the limits of performance.

I. The Need for FLOPS

THIS paper is concerned with the use of computers for the "simulation of physics on a computer." Simulation involves understanding the behavior of an object that can exist in the physical space of our universe and is the largest but not the only use of supercomputers today (major exceptions being databases and the use of computers for cracking cryptologic codes).

There have been a variety of efforts in the last couple years to identify future needs for large supercomputers, some of which are plotted in figure 1. In general, these efforts reveal a continuum of applications weighted toward low performance, but including key applications at about 10^{21} FLOPS (1 Zettaflops).

A. The SCaLeS workshop and report [SCaLeS 03, and plotted in figure 1] reviewed 10 applications areas, seeking and finding support for valuable science from supercomputers 100× and 1000× today's performance (or 100 Teraflops and 1 Petaflops).

B. NASA performed a study [NASA 99, and plotted in figure 1] of the speed needed in a future supercomputer that would permit it to compute fast enough that an engineer operating it would not have their thinking impeded by supercomputer slowness. This report concludes the computer

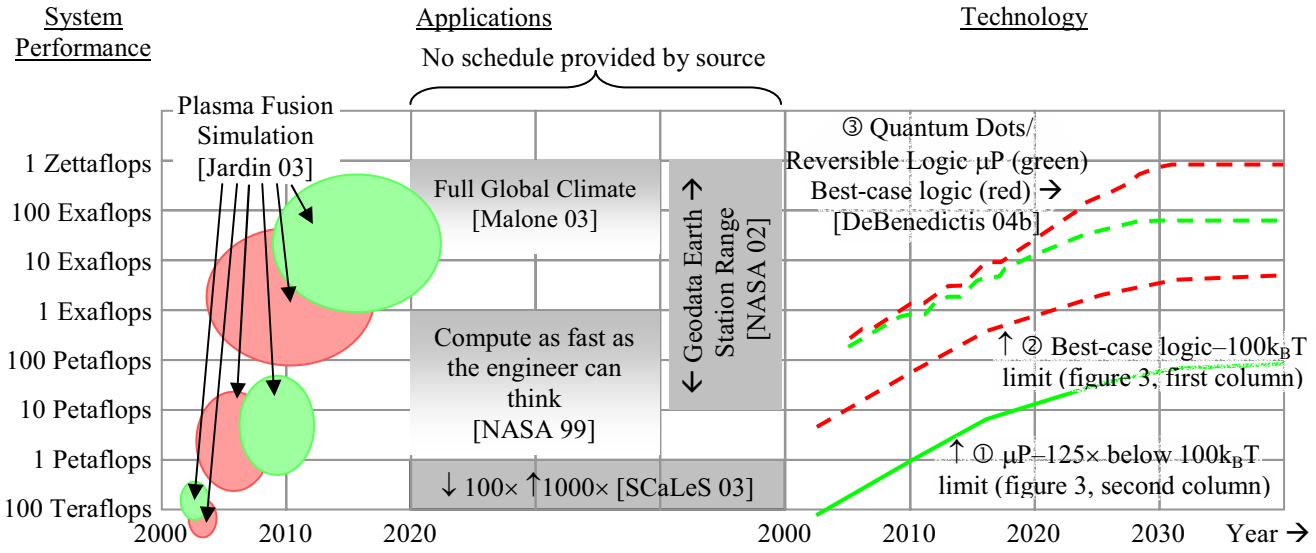

Figure 1. Supercomputer applications and technology projections.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

should be about 10^{18} FLOPS (1 Exaflops). This resource would be dedicated to a single engineer at a time, meaning the computer center would need many times this power to serve many engineers.

C. Research toward the generation of electric power by controlled nuclear fusion often rates as the second largest potential consumer, peaking at 3×10^{20} FLOPS in [Jardin 03, and plotted as ellipses in figure 1]. These are simulations of Hydrogen-containing plasmas at temperatures high enough for nuclear fusion.

D. The processing of Earth climate data appears to be the FLOPS leader at present. These are efforts aimed at collecting and storing worldwide information on the Earth, such as atmospheric, oceanic, and biological information collected by satellites and using this data as boundary conditions for forecasting future weather and climate. The SCaLeS report [Malone 03, and plotted in figure 1] gives a peak FLOP rate of 10^{21} FLOPS (1 Zettaflops, but actually a range of 10^{20}-10^{22} FLOPS). A NASA report [NASA 02, and plotted in figure 1] gives a range of 10^{16}-10^{21} FLOPS.

II. Defining Simulation of Physics on a Computer

The applications defined above use a variety of computational methods, such as Finite Difference, Finite Element, Particle-In-Cell, etc., but many of these fall within the overall method of "simulating physics on a computer" very eloquently defined by Richard Feynman [Feynman 82]. This class of applications places specific demands on the architecture of the underlying computer. Figure 2 illustrates the division of space into a series of regions, or cells. Each cell holds the state of a particular region of space, such as the particles in a fusion plasma or the composition and motion of air or water. The computer simulates the evolution of space by updating the state of each cell for an interval ΔT, based on the computer evaluating the applicable laws of physics.

This general class of calculation consumes resources on the computer in a particular ratio. Specifically, during each ΔT time step, the laws of physics are evaluated for the entire

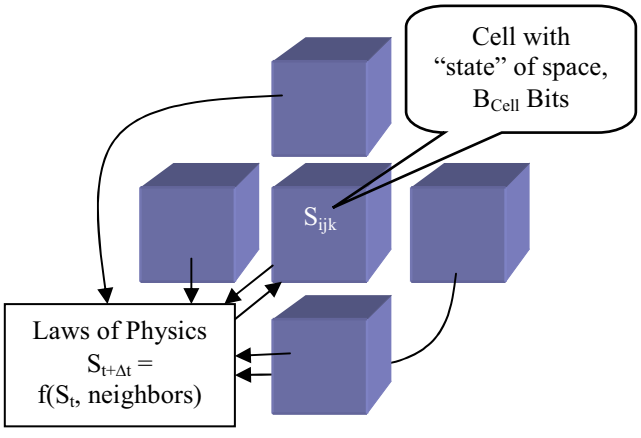

Figure 2. Simulation of Physics on a Computer. Each cell is comprised of B_{Cell} bits of computer memory representing the state of spatial area or volume in the problem. The computer updates the contents of each cell repeatedly for time intervals ΔT.

contents of the state memory. These evaluations will require access to the state in neighboring cells in accordance with the underlying geometry of the problem (i. e. a 3D simulation will require "nearest neighbor" communications between cells with "nearest neighbor" being defined in accordance with a 3D layout).

Reality is slightly more complex, however. Real algorithms need to identify when the calculation completes, adjust ΔT, etc. This step varies by algorithm, but typically involves a simple calculation (like addition or max) over some parameter of the entire simulation space. The MPI Allreduce function [MPI web] is a popular way of doing this and will be used as an example in this paper. As illustrated in figure 3, a more complete description of the calculation involves processors evaluating the laws of physics on cells, followed by a communications event across the entire machine.

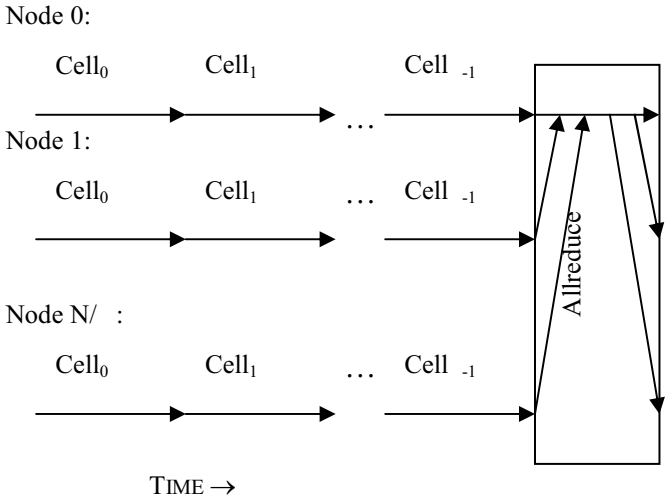

Figure 3. Organization of a Time Step. Each computer "node" evaluates the laws of physics for a group of cells. Subsequently, all the nodes in the application synchronize and exchange data about ΔT and/or termination of the algorithm.

Some problems are further complicated by an "outer loop" for optimization or iteration. A problem may be to find the optimal solution for something, such as the human change to the Earth that would best mitigate global warming. A typical calculation of this sort involves many repetitions of the calculation described above with little interaction between the repetitions. The lack of interaction makes architectural issues more straightforward.

III. Quantifying the End of the Current Trend for Supercomputer Performance

The issue of the "End of Moore's Law" has been extensively studied in general and the results are quite voluminous. However, supercomputers doing "simulation of physics on a computer" stress the underlying technology in a specific way and lead to a readily understood and intuitive limit. The limit for supercomputing is that the system consumes progressively more power until the operator can no longer afford the electric bill.

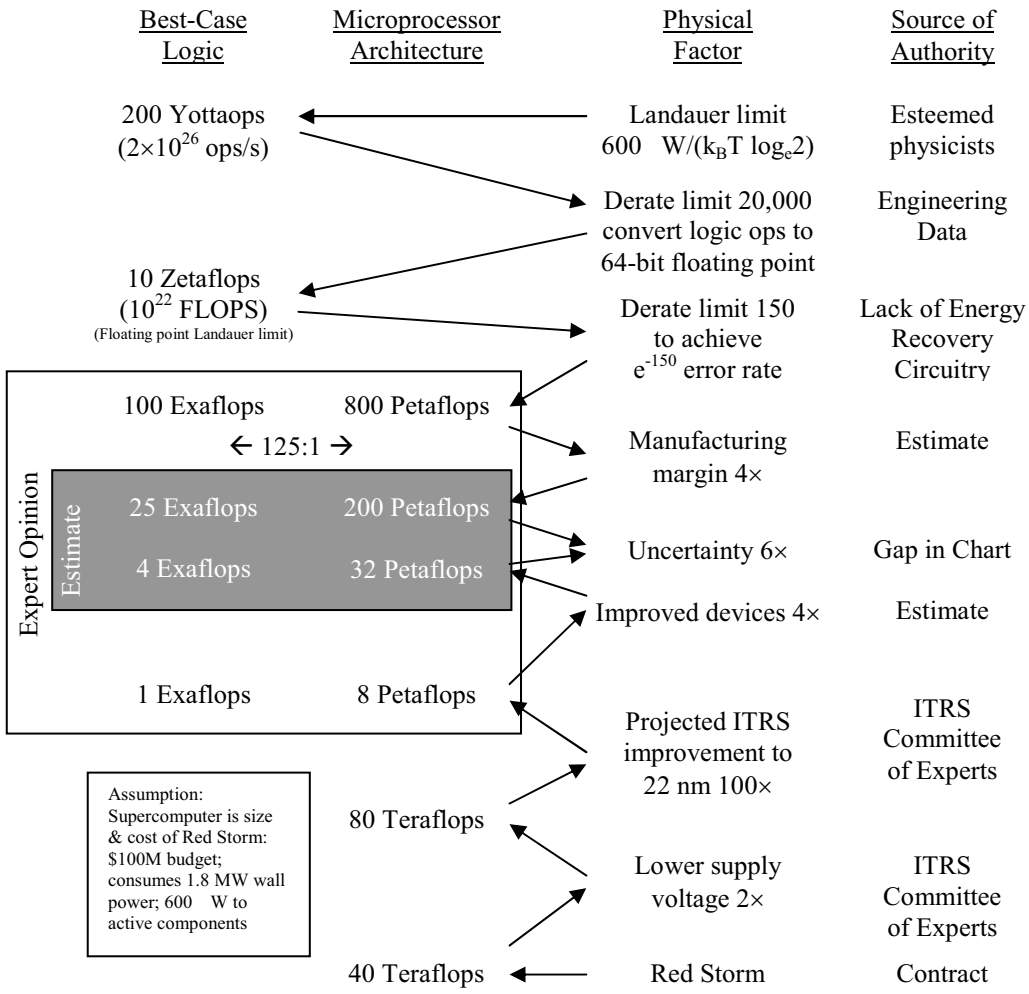

Figure 4. Limits on Supercomputers Set By The Laws of Physics. This chart derives the upper bounds on performance be derating the physical limits while simultaneously building up possible performance from known supercomputers and industry plans. A small region of uncertainty appears at the center.

Irrespective of device miniaturization, the type of logic in use today requires a minimum amount of heat generation to avoid spontaneous glitches due to thermal noise. The attribute of computer logic is not new; von Neumann understood it at the time the computer was invented [von Neumann 56]. We expect computer logic not to make spontaneous errors (glitches). Furthermore, the consequence we impose for a glitch is that we replace the computer. Since computers have a finite life expectancy, this suggests that the probability of a glitch be less than one in the total number of logic operations the computer will perform in its lifetime. A 100 Exaflops supercomputer expected to run ten years without error would require a reliability of one error in about 10^{33} operations. To avoid over specificity, let us just say reliability must be less than one glitch in 10^{30}-10^{40} operations.

The experience we will have with semiconductor reliability over the next dozen years is similar in many ways to driving out of town in a car while listening to FM radio: as we drive further away from the radio station, the initially clear signal acquires a "hiss" which grows over time until it obscures the signal and we turn the radio off.

This noise comes from the first amplifying transistor in the FM radio: this transistor is exposed to both the radio signal from the antenna and the thermally induced noise signal from the vibrating electrons in its own structure. The antenna signal weakens as we drive out of town, causing the noise signal to grow in proportion.

The transistors in a logic gate are similarly exposed to the signal from the preceding gate and thermal noise from their own electrons. While the magnitude of noise in logic gates is exactly the same as the noise in FM radios (its magnitude is k_BT, where k_B is Boltzmann's constant of 1.38×10^{-23} and T is the temperature in elvins), Moore's Law is causing the signal energy to decline exponentially with time (through subsequent generations of electronic technology).

Logic gates are constantly comparing their input voltages against a threshold to determine whether they are receiving a "0" or "1." The effect of noise is nil unless the noise signal makes an excursion in the opposite direction of the logic signal sufficient to exceed the threshold separating the logic levels. The probability of this occurring grows exponentially with the power of the noise signal. While the probability of

misinterpreting a bit is dependent on many factors related to circuit design, a rule of thumb is to assume it will be $e^{-E/kT}$ (see [DeBenedictis 04a] for a full treatment). By this standard, a switching energy should be about $100k_BT$ to meet the requirement of less than one glitch in 10^{30}-10^{40} operations, per above. (By comparison, today's circuits operate at about $100{,}000k_BT$ for a vanishingly small error rate of $10^{-43{,}000}$.)

Figure 4 derives the limits of a "normal size" supercomputer based on current technology from two complementary directions. The author's model of a "normal size" supercomputer is the ASCI Red Storm system at Sandia, but Red Storm is of similar size to a "leadership" class supercomputer in a US supercomputer center: $100M construction budget, taking 1.8 MW electric power from the wall with 600 W going to the active components. In figure 4 a physics analysis goes from the top down and a semiconductor industry projection goes from the bottom up. The two meet in the center with a small but instructive gap.

The limits of computer logic (known as "irreversible logic") have been known from the first days of computing. Von Neumann is generally credited with inventing modern computer logic, and it is clear that he understood its limits. However, a contemporary of Von Neumann's, Landauer [Landauer 61], identified that these limits applied only to "irreversible" logic as opposed to logic in general. The $k_BT \log_e 2$ limit defined by Landauer yields the top number in figure 4 of 150 YOPS (YottaOPS, Yotta is the SI prefix for 10^{24}).

However, today's supercomputing is based on floating point not logical operations. A double precision operation in today's logic if formed from about 20,000 logical operations, given a reasonable mix of adds and multiplies. This yields the second number in from the top in figure 4, 10 ZFLOPS (Zetaflops, Zeta is the SI prefix for 10^{21}).

An analysis of trends in the semiconductor industry proceeds up from the bottom of figure 4 and is based on the Semiconductor Industries Association's (SIA's) International Technology Roadmap for Semiconductors (ITRS) [ITRS 02] and summarized in table I. This is a study published each year setting goals for up to a dozen years in the future. We will start the upward extrapolation with the Red Storm system at Sandia, although most modern microprocessors would yield a similar result. Red Storm is built from 130 nm semiconductor technology and will achieve 40 Teraflops by purchase contract. The ITRS predicts an increase of 2× and 100× between 130 nm and the emergence of 22 nm technology in 2016. This yields a preliminary peak of 8 Petaflops.

However, the physical limits analysis presumes the theoretical best-case logic, or the logic with the best performance over the set of all possible arrangements of gates and transistors. By contrast, the ITRS analysis extrapolates the Red Storm system as designed with Opteron microprocessors. The advantage of a microprocessor over best-case logic is that it can be programmed after fabrication to address a variety of applications. One cannot say that the efficiency of best-case logic is fundamentally more important than the flexibility of a microprocessor or vice versa, so figure 4 has separate columns for the two approaches. The left column provides the performance limit for engineers willing to design their own custom logic, whereas the right column is the limit for programmers who wish only to program a microprocessor. Graphs ① and ② in figure 1 also illustrate the limits of microprocessor and custom logic: these graphs illustrate project performance improvements from ITRS projections but "flatlining" as the asymptotically approach the limits. It would be possible to design a "low power" microprocessor that bridges the gap between the columns [BGL web, Davis 04].

TABLE I
PROJECTIONS OF SEMICONDUCTOR PROPERTIES

Year of Production	2010	2013	2016
DRAM ½ Pitch (nm)	45	32	22
MPU/ASIC ½ Pitch (nm)	50	35	25
Physical Gate Length high-performance (HP) (nm)	18	13	9
Power-delay product for (W/L_{gate}=3) device [$C_{gate}*(3*L_{gate})*V_{dd}^2$](fJ/device)	0.015	0.007	0.002
Static power dissipation per (W/L_{gate}=3) device (Watts/device)	9.70E-8	1.40E-7	1.10E-7
High-performance NMOS device t ($C_{gate}*V_{dd}/I_{dd}$-NMOS) (ps)	0.39	0.22	0.15

White – Manufacturable solutions exist and are being optimized	
Yellow – Manufacturable solutions are knows	
Red – Manufacturable solutions are not known	

Figure 4 is thus left with a gap of two orders of magnitude representing the uncertainty in the opinions of experts on the potential upper limit on performance of supercomputers (i. e. 1-100 Exaflops for best-case logic and 8-800 Petaflops for microprocessors – all applying to a supercomputer the size of Red Storm). While this gap is fairly wide, it is unlikely that the real limit will be at the wide extremes:

All technologies require some tolerance for manufacturing variances, inefficiencies in wires, noise margins, and so forth. We will derate by 4× in this paper. (The only similar analysis I have found [Frank 99] used a derating factor of 12× However, this analysis was seeking an "expected value" rather than a "limit" and so it is generally supportive of 4× as a limit.)

Furthermore, the ITRS is only projecting the state of semiconductors in 2016, not the end of Moore's Law. We similarly find no expert opinion on semiconductor progress beyond 2016, but let us guess it can deliver another 4× in performance improvement.

While unverified by any authority other than the author, the two assumptions cut the gap to 6×. Given historical gains in computers over the years, a performance uncertainty of 6× for the end point is small.

Figure 4 has direct applicability to the application examples presented earlier in this paper. To be specific, 100× and 1000× gains over today's supercomputers are easily within bounds. However, a 1 Exaflops system for engineers [NASA 99] exceeds the limit for microprocessor-based solutions but might be possible with some advanced architecture. The Zetta scale

requirements for climate modeling and fusion simulations are around 100× above the limit for special purpose logic (10 Exaflops×100 = 1 Zettaflops). It should be noted that figure 4 is relative to the size of the user's budget (taken as $100M) and can be raised by a proportionately larger budget. Thus a Zettaflops system based on custom logic should be possible for 100×$100M, or $10B.

IV. Beating The Limits of Current Technology

Figure 4 does not represent the end of progress in computing, but it merely the local maximum associated with today's popular technology. There has been an extensive body of knowledge developed in the last several decades under the title "physics and computation" where the limits represented by figure 4 have been circumvented by clever developments. A full treatment of "physics and computation" to improve supercomputers is out of the scope of this paper (although discussed elsewhere [LACSI 04]), but is still relevant. This section will review and reference some of the technology components that would enable computing up to the Zettaflops level. However, I expect the reader to see that developing these technologies sufficiently would be a very expensive proposition. In the author's view, such a large development expense would not be funded until there is a thorough review of the limits of current technology – which is the objective of this paper.

Curve ② of figure 1 is the natural performance curve that will result from Moore's Law running its course. This is the curve of performance resulting from logic gates of the current design but with faster transistors. In current design 1's and 0's needed for the internal operation of the computer are created by drawing charge from one of the power supply rails. When the bit is no longer needed (such as due to its arrival at the other end of a wire), the signal is destroyed by releasing the charge into the other power supply rail with the entire switching energy being turned to heat in the process. As has been described earlier in this paper, the amount of energy must be greater than about 100 k_BT for the computer to be reliable. This mechanism is the predominate source of power usage in today's computers.

While the 100 k_BT energy limit is unchallenged, it is possible to "recycle" most of this energy. The approach is to power circuits not through a DC voltage but an AC clock signal generated by a resonator. The 1's and 0's are created by drawing 100 k_BT energy from the resonator as it swings to one extreme. However, when a resonator reaches it limit in one direction, it also reaches the maximum of some force that will pull it back to the center and then to the other limit. Thus, most of the 100 k_BT energy put into the logic each cycle is "pulled out" by the resonator at the end of the cycle. Resonant circuits can include Inductor-Capacitor ("LC") tank circuits, MEMs resonators [Anantharam 04], and may someday include carbon nanotubes [Legoas 04] (which can oscillate at incredibly high frequencies). Logic families based on this principle have been developed for transistors [Seitz 85, Denker 94] as well as post-transistor devices. Managing energy in this way substantially reduces the heat generated per operation and permits more useful work per watt, but it comes at the cost of adding resonators to chips and redoing all logic designs from DC to AC power.

The "recycling" above is limited by an unavoidable transformation of information into heat when information is destroyed [Landauer 61], and logic gates in use today (such as AND, OR, NAND, and NOR and known as "irreversible logic") destroy information when they convert two or more inputs into a single output. The minimum energy for an irreversible logic gate is on the order of $k_BT \log_e 2$, or 150× below the 100 k_BT limit discussed previously. This is known as the Landauer limit and is illustrated in figure 4.

Even the Landauer limit can be beaten with more ambitious changes. Since the logic gates in use today require a minimal heat generation per the laws of physics, dropping below this heat level necessarily involves changing logic gates. There is a sizeable body of literature and some prototype circuits using "reversible" logic gates. These are gates with the same numbers of inputs as outputs and which do not destroy information. For example, a Fredkin gate [Fredkin 82] with 3 inputs and 3 outputs simply exchanges two inputs in response to the third, yet has been shown to be a complete logic family. These principles of logic have been used to create arithmetic [im 01], microprocessors [Vieri 99], memories [Vieri 98], and a "C" like programming language [Frank 97b]. However, the computers are somewhat different from those today. For example, floating-point operations destroy information as a part of their normal operation: in aligning the operands for a floating-point add, the low bits of one of the numbers are shifted away and lost. Thus, there is no way to create a floating-point operation from gates that do not destroy information. However, there are methods for architecting a computer where whole calculations are done and then "undone" in order to restore the computer to the original state [Bennett 89].

The techniques outlined above may require a post-transistor switching device to provide real benefit. The author has searched diligently and unsuccessfully for a researcher that can claim or demonstrate that the benefits from the techniques above overwhelm various forms of added overhead. However, if one is willing to embrace post transistor devices such as Quantum Dots [Timler 03], Y-branch switches [Forsberg 03], Rod logic [Drexler 92], Helical logic [Merkle 96], or a dozen others, the performance ceiling rises considerably. Graphs ③ in figure 1 project performance for a system based on quantum dots in conjunction with all the methods given above [LACSI 04]. These technologies reach the Zetta scale as required by applications, although they would have an impact on the design of computers reaching from the devices to software.

V. Can We Reach The Limit?

Just because I have shown the laws of thermodynamics would not be violated by a supercomputer reaching the limits in figure 4, it remains to be shown how to reach this limit with known architectures being applied to the problem illustrated in

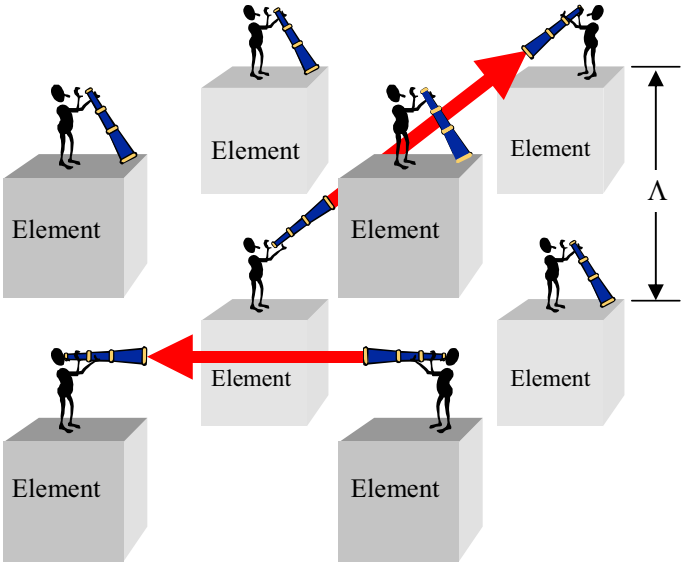

Figure 5. Aerogel Computer Model. Cells represent either gates or memory locations. Cells communicate through optical pulses that travel at the speed of light without interference.

figure 2. My approach is to compare the modeled running time of an application on two hypothetical computers. The modeling builds on work in applications modeling [Christopher 04a, Christopher 04b, Hoisie 00, erbyson 01, Petrini 03], or the prediction of the running time of applications on computers. One of these computers is the "Aerogel" computer model, or a model that can meet the maximum performance possible for any computer given the laws of physics. (The Aerogel model was developed by the author and appears to be unique, but others have sought computational models for similar purposes and come to comparable ends [Frank 97a].) The other is a fairly realistic model of a computer that would be constructed with integrated circuits available at some point in the future in accordance with semiconductor industry plans. If the performance of the realistic computer model is close to the "Aerogel" computer, we can conclude that we will be able to approach the limits of computing with technology we understand.

Figure 5 illustrates the Aerogel computer model. A computer in this model is made of a series of elements, packed into a rectangular array on pitch Λ and each of which may contain a logic device or a bit of memory (thus being equivalent to one transistor). Λ is initially set to the cube root of the volume of a nominal logic transistor or DRAM cell (I am not too precise about the initial Λ because it is increased later, see below). An element with a logic gate is presumed to have a propagation delay τ and dissipate a certain amount of energy E_{Gate} every time it switches. The wiring in the model is via congestion-free channels that move data at the speed of light. This is illustrated in the figure by telescopes pointing in whatever direction is necessary and pass data via light pulses.

TABLE II
COOLING SYSTEMS

Method	C_x Capacity
Air	47 W/m^2
Water	63.7 MW/m^2
Fractal Cooling[a]	1 GW/m^2
Pulse[b]	∞

[a]Submicron ice particles encased in a polymer in hexane. The ice melts as the device is cooled. Theoretical study by [Drexler 92]. Quoted as 100 W/cm^2. [b]The theoretical limit for a system that is operated intermittently. Equivalent in calculations to an infinite cooling capacity.

Since the model is theoretical, programming is accomplished by a pencil and paper analysis. To program an Aerogel computer, each cell is designated to be either part of a logic gate or a memory cell. The interconnect is likewise programmed by designating the pointing direction of the telescopes. With this programming in place, one can calculate the running time for an application through applications modeling as a function of the speed of light delays and propagation delays.

If we perform such an analysis using projected values for E_{Gate} and Λ spacing values corresponding to real transistors, we find that a computer will overheat and destroy itself in microseconds. To remedy the analysis, I specify that the computer in figure 5 is to be "pulled apart" or linearly expanded until the entire computer has sufficient surface area to be cooled. If one were to observe the resulting computer, it would be a series of transistors suspended in empty space with signals flowing via light pulses (or wires) between cells. If such a computer were actually constructed, it would be similar to an Aerogel (hence the name of the model).

There are various cooling technologies available in practice, differing by the amount of heat that can be removed per unit of surface area. Table II is a very brief overview of the practical options considered in this paper. One must specify the capacity of the cooling system (in units of E_{Gate} that can be removed per square area of surface) in order to know how much to inflate the computer. Of course, the amount of inflation effects running time due to increased distances signals must travel at the speed of light. As a consequence, the running time will depend on the type of cooling specified.

The running time of an algorithm on an Aerogel computer is essentially a measure of the algorithm's complexity as determined by the laws of physics. To elaborate, computer science defines the "complexity" of an algorithm as the asymptotic dependence of running time on the size of the problem. The running time on an Aerogel computer is similar in some ways, but is additionally specified to an actual number (instead of merely the asymptotic dependence). Furthermore, the Aerogel computer model is much closer to physical reality than the model used in complexity theory: The Aerogel model includes the effects of the speed of light, cooling, and propagation delay whereas complexity theory merely counts operations. However, in spite of these differences, the formula for running time on an Aerogel computer is metric for the quality of an algorithm. We call the running time a "complexity metric."

VI. An Example

We will develop the equation for applying the Aerogel model to an application as shown in figure 2 and defined by $C_{Physics}$, $T_{Physics}$, and $E_{Physics}$. We will also assign specific values to these parameters based on a popular supercomputing application and plot the results.

The prototype application is a shock hydrodynamics application with the obscure name "CSQ to the Three Halves," but widely known by its acronym CTH [CTH web]. CTH models moving materials, such as a bullet striking a target or gasses colliding at supersonic velocity. It was developed at Sandia and is reportedly the most popular supercomputer application for the Department of Defense (DOD). The results of this analysis are plotted in figure 6.

The CTH program gives insight into practical parameter values. Parameter values depend on the number of materials being simulated, but we will restrict this analysis to a problem with two gasses:

- Two gas problems use 240 bytes/cell. Thus, we use B_{Cell} = 240 bytes = 1920 bits.
- The number of floating point operations to evaluate the physics (called the "grind time") has a mean of about 3500 FLOPs and a standard deviation of 3500 FLOPs. We use these figures [Brightwell 04]. Thus we use MEAN$\{T_{Physics}\}$ = 3500×200τ, σ = 3500×200τ $C_{Physics}$ = 100,000 elements, and $E_{Physics}$ = 3500×20,000 E_{Gate}.
- The CTH program exchanges all boundary cells (all 240 bytes) 11 times during each time step. Thus, the bandwidth per iteration will be 11×B_{Cell} =21 bits/cell/cycle.

We will assume the applicable laws of physics per figure 1 can be evaluated by a physics unit comprised of $C_{Physics}$ elements in time $T_{Physics}$, producing $E_{Physics}$ heat. To program this part of the Aerogel model, one would create a schematic diagram of floating-point adders, multipliers, etc. to evaluate the equations for the laws of physics in a manner similar to a signal processor. The equivalent network of gates would then replace each adder, multiplier, etc. The gates form the cells of the Aerogel model and the wiring translates into the directional pointing of the telescopes.

Let us define a node as comprising a physics unit from the paragraph above, memory to hold cells = × B_{Cell} bits worth of state, and some accessing logic as described below. We will also assume $\sqrt[3]{}$ N/) is an integer, so that each node can simulate a cubical sub region of equal size. In figure 6, each element comprises one floating-point number of 64 bits, making the memory 64 cells in size.

We describe elsewhere [DeBenedictis 04a] how to construct a memory accessing system that is quite efficient compared to the physics unit and can be neglected. To be specific, the access pattern for the problem in question is entirely deterministic, permitting the use of a sequence counter comprised of ~$\log_2$ flip flops and a handful of gates. A deterministic access pattern substantially relaxes any "memory latency" constraint because memories can be prefetched as far ahead as is convenient. Architectures exploiting memory access pattern regularity have been explored elsewhere [Sair 03]. Our conclusion is that delay time can be neglected due to overlap with other activities and that the number of cells and their power consumption will be less than a floating point unit. Assuming the problem involves floating point, this permits us to ignore the access logic without fear of changing the result of the analysis.

However, it is worth noting that the result of all this is very similar to a traditional vector floating point unit.

The computation time for a single time step is given below as the time for each node to evaluate its cells plus the time for the global communications step, designated $T_{Allreduce}$ and described later:

$$T_{Step} = \times T_{Physics} + T_{Synch} + T_{Allreduce}$$

The total number of elements in the supercomputer is given by

$$T_{Supercomputer} = B_{Cell} \times N + \frac{N}{} \times C_{Physics}$$

The parameter T_{Synch} represents waiting time due to the fact that $T_{Physics}$ is a random variable and some nodes will take longer to execute their collection of evaluations than others.

$$T_{Synch} = \Phi^{-1}(1\text{-}1/(N/ +1)) \times \times T_{Physics}$$

where Φ^{-1} is the inverse of the cumulative normal distribution.

The time for the global communications step will be bounded from below by the time for a signal to traverse the physical structure of the supercomputer, given by

$$L_{Edge} = \sqrt[3]{T_{Supercomputer}} \times \Lambda$$

$$T_{Allreduce} = \frac{\sqrt{3} \times L_{Edge}}{c}$$

The equation above has proven to be controversial. Given that $T_{Supercomputer} \propto N$, a number of parallel computer advocates have objected to my claim that the optimal Allreduce running time is $O(\sqrt[3]{N})$ whereas they know of parallel computer algorithms that are O(log N). These views can be reconciled by the difference between the physically accurate Aerogel computer model and the mathematically abstract parallel computer model.

Allreduce can be performed on a parallel computer by forming a binary addition tree of nodes, adding values from the nodes, and then broadcasting the result back using the same tree in reverse. Since the parallel computing model counts only sequential steps, the summation takes 2 $\log_2 N$ steps.

However, the Aerogel model also counts the cost of communications. As N increases, the size of the computer increases as $\sqrt[3]{N}$ because the memory cells have finite size and must be packed in the three dimensions of the universe we live in. The communications time for Allreduce can never be less than the time required for a signal to cross the supercomputer when traveling at the speed of light, and this time is proportional to $\sqrt[3]{N}$.

So, what would happen if my critics tried to build a series of progressively larger parallel computers and then tried to measure Allreduce timing in order to verify the O(log N) running time? All the machines in this series would specify

the same message passing latency between arbitrary nodes in the system. As machines become larger, the engineer would find progressively less time for the router to switch messages after the speed of light delay in the cabling was subtracted from the message latency. Above some size, the routing time would become negative and it would not be possible to build the machine. A similar experiment with an Aerogel computer would not have this problem.

However, what about the time to perform the mathematics of reduction? In this case, the reduction is addition. Floating point formats can be designed such that floating point comparisons can be done with the same logic as integer comparisons. Furthermore, integer comparisons can be done in bit serial order, most significant bit first. This makes the calculation time negligible compared to the communications time [DeBenedictis 04a].

The final constraint relates to cooling. The equations below state that the heat flux that can be removed from the surface of the machine exceed the machine's wattage

$$6 \times C_x \times L_{Edge}{}^2 \geq \frac{N \times E_{Physics}}{T_{Step}}$$

We wrote a computer program to create a series of hypothetical supercomputers, each optimized to produce the best running time in accordance with the equations above. Figure 6 shows the output of this program, plotting the running times of various families of optimized supercomputers as a function of the memory depth .

The program explores supercomputers running one iteration in accordance with equations given previously with N=n×n×n for n=50,000 cell space, where the cells are distributed evenly onto N/ nodes. The supercomputer is a 3D solid pack of nodes when program explores families built from the Aerogel model (although inflated to meet cooling limits). Supercomputers are an assembly of chips of one or more nodes when the program explores realistic families.

The program assumes same transistor specifications for both Aerogel and Realistic families. These come from the ITRS, summarized in Table I. The ITRS includes separate transistor specifications for high performance, low power, and memory (not shown in Table I), and the program uses these various transistors as appropriate.

For the realistic family only, the program uses a maximum chip capacity and a maximum I/O bandwidth from the ITRS. Furthermore, the realistic model assumes a 3D packing of chips but where the machine volume per chip cannot be less than some fixed volume set by the volume of a chip plus heat sink in a standard configuration.

The program's output is constructed as follows: The program separately explores Areogel and realistic computer families, plotting the results in figure 6 with thin and thick lines. The program separately explores cooling by air, water, fractal plumbing, and by intermittent operation, with cooling capacities defined in Table II. The different cooling technologies are plotted in different colors. The program sweeps parameter to form the horizontal axis.

For each computer, the program optimizes the number of

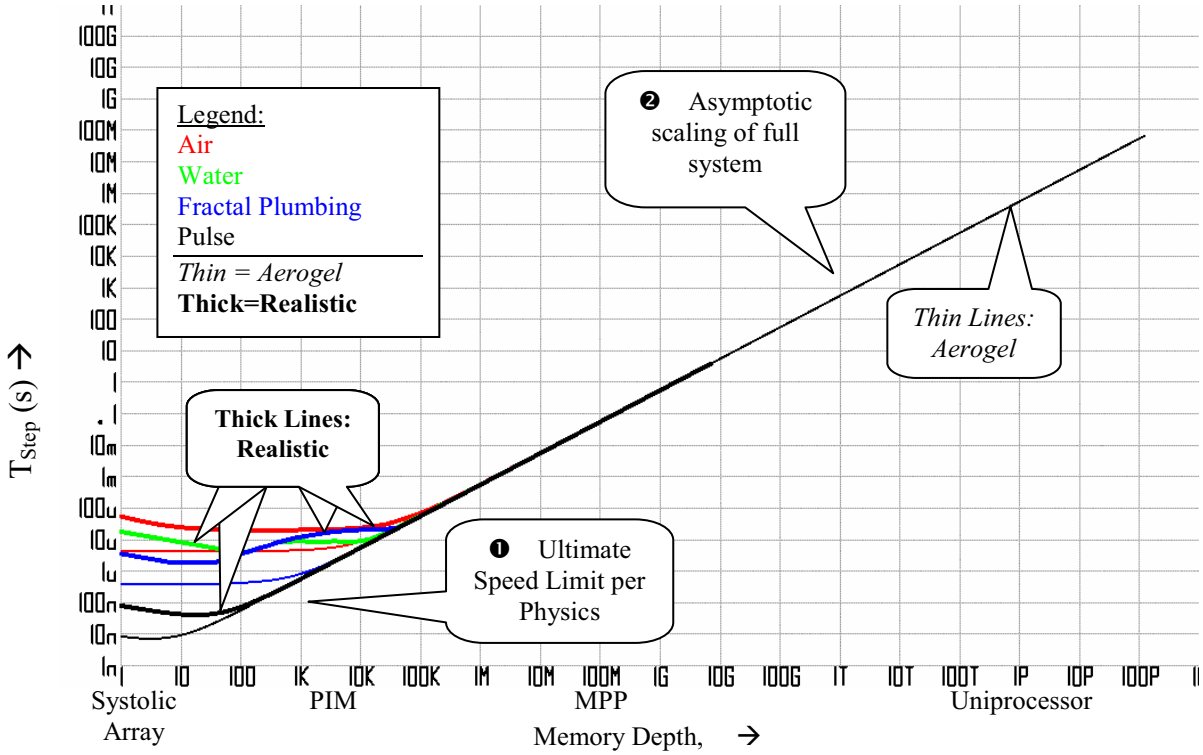

Figure 6. Results of Applications Modeling on Aerogel and Realistic Computer Models. Horizontal axis is the number of cells per "node," representing systolic arrays, PIMs, MPPs, and uniprocessors. The graphs show execution time per time step (lower is better) for various type of cooling technology. Thin graphs are for Aerogel computer and think ones are for realistic model. Note Aerogel and realistic are the same except at the chart's left. Also, air-cooling is worst, but not by much.

nodes per chip. While the number of nodes (each holding cells) is obviously limited by the maximum transistor capacity of a chip, there are other considerations as well. The chips may be deliberately under filled to avoid overheating or if I/O bandwidth becomes insufficient and would cause a bottleneck. The program performs this optimization by sweeping the number of nodes per chip and doing a performance model for each combination.

For each candidate computer, the program finds the smallest Λ above minimum device sizes for which the system does not overheat. Inflating Λ has two effects, both of which decrease power consumption: it increases the surface area for heat outflow and decreases the speed (due to increased signal travel distance). These factors are monotonic but one is nonlinear, so iteration is required.

The basic performance modeling code calculates the time step execution time and power consumption given all the assumptions above. The time step time will be a local compute time, a bandwidth component for "surface nodes," and an "Allreduce" time dependent on the overall size of the machine.

Figure 6 shows results from solving these equations. More specifically, the thin lines in figure 6 are the runtime of the Aerogel model using transistor parameters from Table I and the equations in the text of this article (the source code for the equations plotted is available in [DeBenedictis 04a], and are somewhat more detailed than the equations in this article). The thick lines in figure 6 are the result of a more realistic model. The realistic model uses the same basic transistors, but the transistors have "leakage" and are restricted to being on chips with heat sinks and limited pin bandwidth in accordance with the 2016 ITRS specifications [ITRS 02].

Figure 6 shows the predicted running time per time step of the CTH algorithm for various computers. The horizontal axis is the number of cells per processor, . The largest values of correspond to a uniprocessor, with the entire problem in memory and a single CPU stepping through the cells one at a time. Values of in the range of 1 billion represent Massively Parallel Processors (MPPs); values in the few thousands represent Processor In Memory (PIM), and values approaching 1 represent systolic arrays or signal processors.

On the left hand side of figure 6 (position ❶), performance is very high due to large numbers of nodes yet ultimately limited by speed of light delays. On the rest of the graph (position ❷), performance falls off due to decreasing parallelism. The higher capacity coolants result in more performance due to the resulting machine being physically smaller and signals having less distance to travel.

Figure 7 is a cost-efficiency analysis of the same computers. This graph plots the number of Teraflops available per dollar spent on a supercomputer per year. The program calculates this cost assuming each chip costs $1000 with 30% of the cost amortized each year, floor space costs $15/ft^2/year, and electricity costs 15¢/ WH.

Figure 7 illustrates the very small advantage of elaborate cooling methods for this problem. The reader will see that the red graph representing air cooling is equal to or lower than any other option. This is because an air-cooled supercomputer must be slightly larger than other more powerful cooling methods. This causes a decrease in performance due to

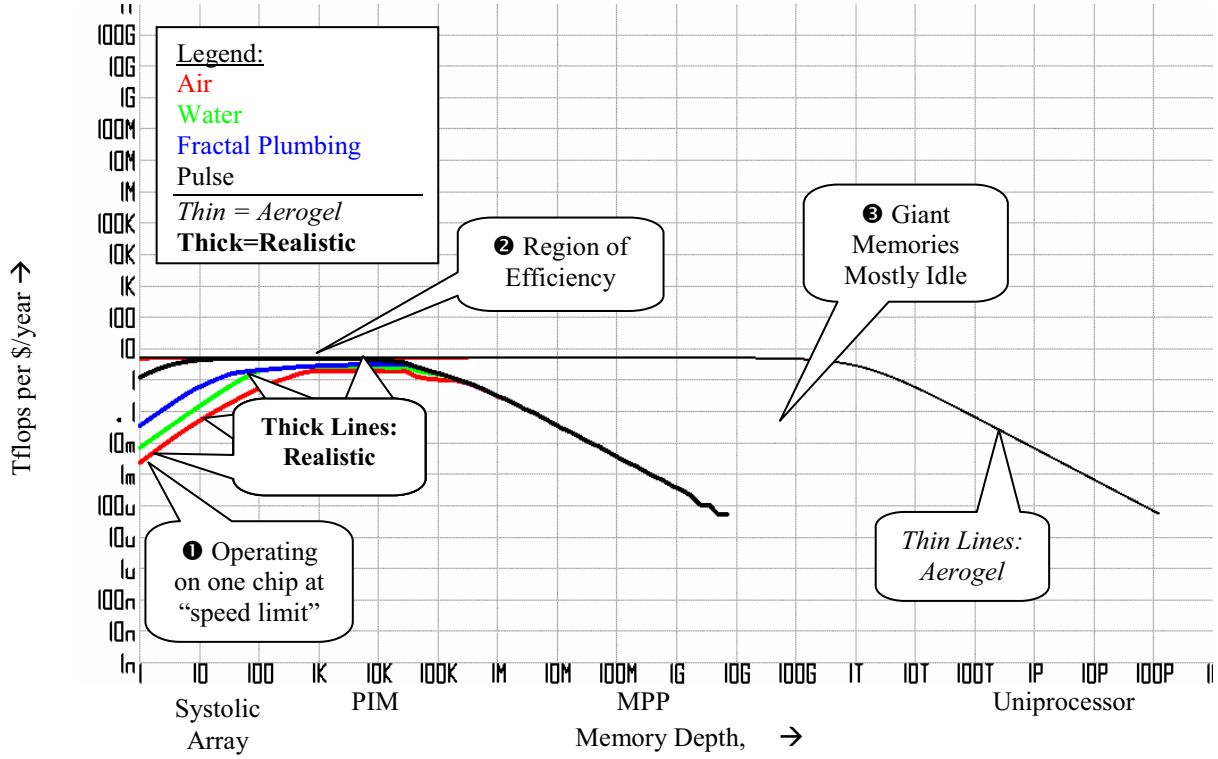

Figure 7. Cost Efficiency of Aerogel and Realistic Computer Models. Graph based on an economic model incorporating cost of equipment, electric power, and machine room space. Graph indicates broad peak of good efficiency.

increased signal travel distance.

By performing this analysis, I claim to have shown that we have the technology to approach the theoretical limit of performance for an irreversible logic computer running the type of problem in figure 2. Through figure 4, I have shown that the semiconductors described in the ITRS roadmap are close enough to ideal that they may be used as a stand-in for ideal with only bounded uncertainty. Figure 6 shows that we understand architecture well enough to exploit these semiconductors to the point of only bounded inefficiency. Bounded means within an order of magnitude.

VII. What Breakthroughs Are Needed?

As an employee for a National Lab and thus somewhat associated with the US Government, the author is interested in knowing what new technologies the Government will need to fund to achieve performance at the Petaflops level and above.

Figure 6 was based on Silicon CMOS technology according to the ITRS roadmap for 2016 [ITRS 02]. It is widely believed that the semiconductor industry will develop this technology for commercial purposes without Government intervention.

Figure 6 assumes embedded memories. The ITRS roadmap predicts that embedded DRAM will be developed commercially for System On Chip (SOC) purposes without Government intervention. The industry is also developing System On Package (SOP) technology (where logic and memory chips are "glued" together to create an effect similar to SOC) [Tummala 99] that may be a suitable alternative.

To achieve sufficiently low signal latency, these algorithms require signals to flow in all three dimensions at the system level. The diagram in figure 8 is the author's depiction of how the necessary structure could be created using commercial parts. Figures 8A and 8B depict a 3D structure comprised of standard PC boards. The PC boards have processor chips and power regulators on one side and memories on the other. The chips then connect through side connectors that are unusual

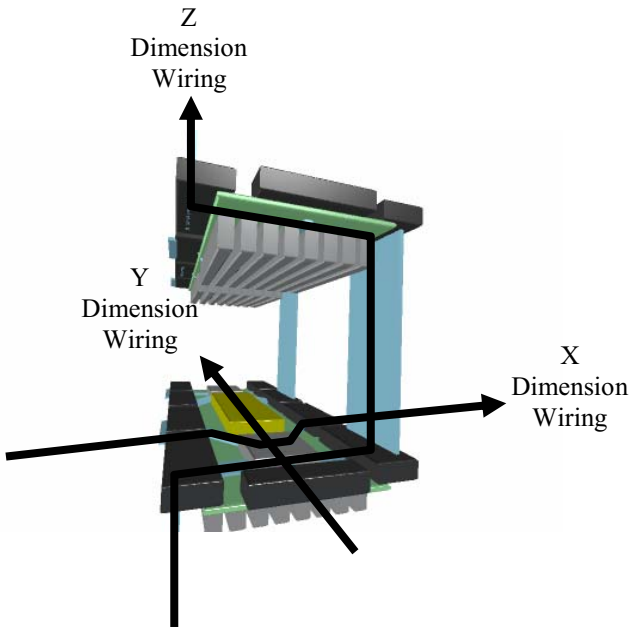

Figure 8A: Mapping of 3D Mesh to Physical Structure

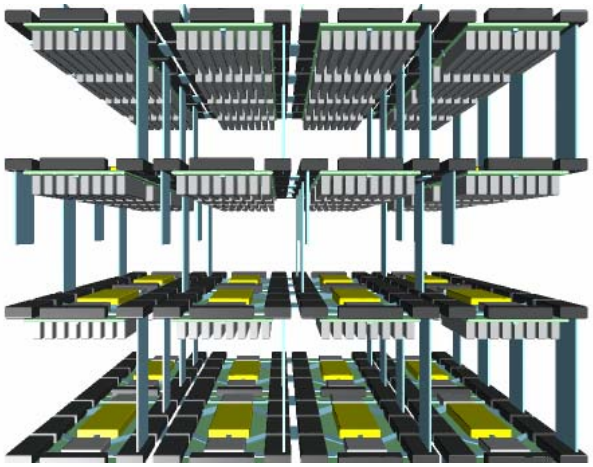

Figure 8B: Three Dimensional Packaging

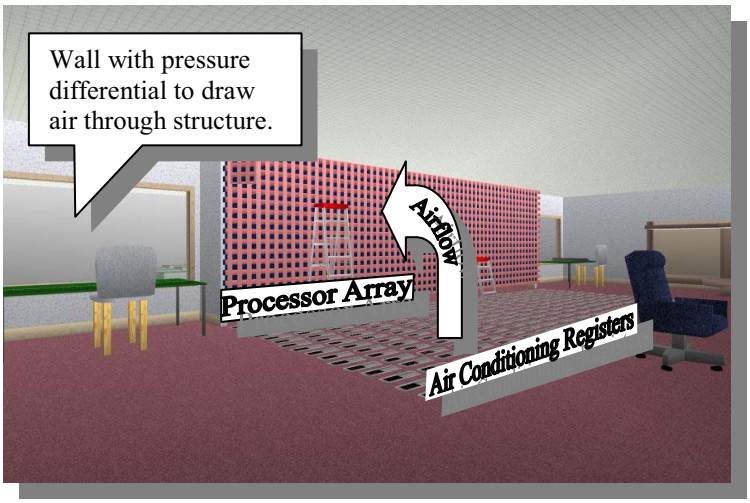

Figure 8C: Air-cooled configuration

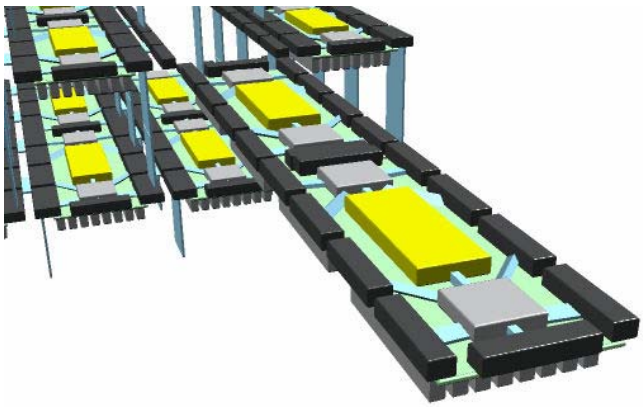

Figure 8D: Serviceability

but commercially used in the Cray 3D and X1 and available commercially from Intercon Systems. Signals can thus flow in all three dimensions. A diagram of a machine with both the necessary cooling structure and 3D signal flow is shown in figure 8C. The grid structures in 8B and 8C are the same but at different scales. The side connectors from Intercon permit disassembly and repair as shown in 8D. The point of this analysis is to report that very little new technology is needed to reach the potentials of supercomputing – at least beyond semiconductors that will be developed anyway.

VIII. The Multi-Architecture Approach

We anticipate a fundamental shift in supercomputer architecture driven by trends independent of supercomputing but to its advantage. As a field, computer architecture was invented in an era when transistors were expensive. As a result, architecture has been seen as a zero-sum game: each architect tries to find the "best" way to organize the gates in a computer so that their architecture can be the one used to build the computer. Due to economies of scale, the microprocessor has emerged as the universal architecture. However, power and cooling are replacing transistor count as the limiting factor on chips and shifting the assumptions that drove the ascendancy of the microprocessor. It is becoming increasingly feasible to put multiple architectures on the same chip, as long as they are not all powered-on at one time. We project a new era where a chip will contain multiple architectures (call each a logic block), each chosen because it is useful for a subset of applications. This approach has been used implicitly by the advocates of many innovative architectures [IRAM 03, Davis 04, Sterling 02, Upchurch 03, Sunaga 96].

Figure 9 illustrates a multiple architecture chip of with no more than a 25% overhead due to the features that create the flexibility. This chip is comprised 75% of Dynamic Random Access Memory (DRAM), which can total multiple gigabytes in a decade or so and plenty for a supercomputer node. Memory consumes chip real estate, but does not consume very much power

Figure 9 also shows three logic blocks, or "architectures." Let us assume that one is a microprocessor and the other two are drawn from the set of popular alternative architectures, such as PIMs, vector units, reconfigurable logic, FPGAs, and specifically logic of the type discussed in this paper. Each of the logic blocks is drawn as a notched square to indicate that it may draw no more than 75% of the power budget for the chip. The power supply (Vdd) is drawn as switched to keep the chip from overheating due to multiple logic blocks being turned on an creating too much heat.

IX. Conclusions

In writing this paper, my objective was to flesh out the widely held belief that Moore's Law would double supercomputer performance every couple years and thereby generate more scientific discoveries, better weapons, or other things of value to society – noting that this analysis has narrowly defined supercomputing as "simulating physics on a computer."

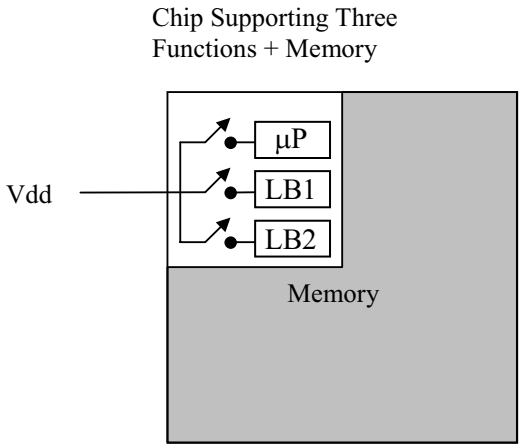

Figure 9. Multi-Architecture. Memory consumes 75% of chip area, but insignificant power. Each logic block consumes 75% of chip's power budget when turned on (only one on at a time) but insignificant area.

The validity of the assertion above depends on whether we are willing to embrace technological change:

If we take a broad view of supercomputing technology, we can find techniques (in the field of "physics and computation") that suggest there is no upper bound to supercomputer performance. There is also a constructive course of action that could plausibly raise supercomputer performance to the Zettaflops level, thus reaching the performance limit of the very most ambitious applications.

However, Moore's Law is strictly defined as a doubling of the number of transistors on a chip every couple years. If we restrict ourselves to advances based on transistors (or current forms of computer logic or microprocessors), we find that the restriction is associated with a lower limit on supercomputer performance that brings us clearly below the level of some applications. However, I showed in sections V-VIII that we can reach these limits with little risk.

While this paper exposes some technical ideas, the ultimate question about how far and how fast to pursue supercomputer technology remains unanswered. I have demonstrated that a natural progression of current technology will raise performance high enough to embrace most of the applications currently envisioned without extensive technology development or building substantially larger machines. Will this be sufficient? Sufficiency would be supported if the applications community were to decide their current projections were too large, our Government finds more money to build large supercomputers of current designs, and no larger applications are found. The reverse could be true as well.

References

[Anantharam 04] V. Anantharam, M. He, . Natarajan, H. Xie, M. Frank, "Driving Fully-Adiabatic Logic Circuits Using Custom High-Q MEMS Resonators," 2004 workshop on Methodologies for Low Power Design, part of ESA '04 (Embedded Systems and Applications). Paper in PDF format at http://www.cise.ufl.edu/research/revcomp/AdiaMEMS/MLPD-04.pdf.

[Bennett 89] Charles H. Bennett, “Time/Space Trade-Offs for Reversible Computation,” SIAM Journal of Computing, Vol. 18, No. 4, pp. 766-776, August 1989.

[BGL web] IBM Blue Gene/L system, http://www.research.ibm.com/bluegene/.

[Brightwell 04] Ron Brightwell, William J. Camp, Ben Cole, Erik DeBenedictis, Robert W. Leland, “Architectural Specification for Massively Parallel Computers -- An Experience and Measurement-Based Approach” To appear in Concurrency: Practice and Experience in 2004.

[Christopher 04a] Thomas W. Christopher, “Radiation Transport Algorithms on Trans-Petaflops Supercomputers of Different Architectures” Sandia National Laboratories Technical report SAND2003-2814, August 2003

[Christopher 04b] Thomas W. Christopher, "Pressures the Radiation Transport Problem Places on Future PIM-Based Supercomputer Designs," Workshop on Software for Processor-In-Memory Based Parallel Systems held in conjunction with the Second Annual IEEE/ACM International Symposium on Code Generation and Optimization, March 21, 2004.

[CTH web] Sandia maintains a Web site for the CTH program: http://www.cs.sandia.gov/departments/9232/cth/.

[Davis 04] ei Davis, Adolfy Hoisie, Greg Johnson, Darren erbyson, Mike Lang, Scott Pakin, Fabrizio Petrini “Blue Gene: A Performance and Scalability Report at the 512-Processor Milestone, Los Alamos National Laboratories unlimited release technical report LA-UR- 04-1114.

[DeBenedictis 04a], Erik P. DeBenedictis, “Taking ASCI Supercomputing to the End Game,” Sandia National Laboratories SAND report SAND2004-0959, March 2004. Sandia technical reports are available by going to http://www.sandia.gov and accessing the technical library.

[DeBenedictis 04b], Erik P. DeBenedictis, “Matching Supercomputing to Progress in Science,” July 2004. Presentation at Lawrence Berkeley National Laboratory, also published as Sandia National Laboratories SAND report SAND2004-3333P. Sandia technical reports are available by going to http://www.sandia.gov and accessing the technical library.

[Denker 94] J. S. Denker. “A review of adiabatic computing,” in 1994 IEEE Symposium on Low Power Electronics. Digest of Technical Papers, pp 94-97, 1994.

[Drexler 92] Drexler, . Eric., “Nanosystems: Molecular Machinery, Manufacturing, and Computation,” John Wiley & Sons, Inc., 1992.

[Feynman 82] Richard P. Feynman, “Simulating Physics with Computers,” International Journal of Theoretical Physics, Vol. 21. Nos. 6/7, 1982.

[Forsberg 03] Erik Forsberg, “Electronic and Photonic Quantum Devices,” Doctoral Dissertation, Department of Microelectronics and Information Technology, Royal Institute of Technology Stockholm.

[Frank 97a] Michael P. Frank, “The R programming language and compiler.” http://www.cise.ufl.edu/~mpf/rc/memos/M08/doc/doc.html. [Frank 97b] Michael P. Frank, “Ultimate theoretical models of Nanocomputers,” Nanotechnology 9 (1998) 162-176.

[Frank 99] Reversibility for Efficient Computing, Michael P. Frank, MIT Ph. D. thesis, 1999.

[Fredkin 82] E. Fredkin and T. Toffoli, "Conservative logic," International Journal of Theoretical Physics, vol. 21, no. 3/4, pp. 219-53, 1982.

[Han 02] Jie Hand and Pieter Jonker, “A System Architecture Solution for Unreliable Nanoelectronic Devices,” IEEE Transactions on Nanotechnology Vol. 1, No. 4 (2002) 201-208.

[Hoisie 00] Adolfy Hoisie, Olaf Lubeck, Harvey Wasserman, “Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications,” in The International Journal of High Performance Computing Applications, Sage Science Press, Volume 14, Number 4, Winter 2000.

[IRAM 03] Overcoming the Limitations of Conventional Vector Processors", C. ozyrakis, D. Patterson. 30th International Symposium on Computer Architecture, San Diego, CA, June 2003.

[ITRS 02] International Technology Roadmap for Semiconductors, http://public.itrs.net. All figures used in this report refer to the ITRS 2002 update.

[Jardin 03] S.C. Jardin, “Plasma Science Contribution to the SCaLeS Report,” Princeton Plasma Physics Laboratory, PPPL-3879 UC-70, available on Internet.

[erbyson 01] Darren J. erbyson, Hank J. Alme, Adolfy Hoisie, Fabrizio Petrini, Harvey J. Wasserman, and Michael Gittings, “Predictive Performance and Scalability Modeling of a Large-Scale Application, , in Proc. of IEEE/ACM SC2001, Denver, November 2001.

[im 01] im, S., Zeisler, C., Papaefthymiou, M., "A True Single-Phase 8-bit Adiabatic Multiplier," in proceedings of the 2001 Design Automation Conference, pp. 758-763.

[ung 82] ung, H. T. "Why Systolic Architectures?," Computer, vol. 15, no. 1, pp. 37-46, 1982.

[Legoas 04] S. Legoas, V. Coluci, S. Braga, P. Coura, S. Dantas, and D. Galvão, “Gigahertz nanomechanical oscillators based on carbon nanotubes,” Nanotechnology 15 (2004) S184–S189.

[LACSI 04] “The Path to Extreme Computing,” workshop in association with the Los Alamos Computer Science Institute Symposium, October 12, 2004.

[Laundauer 61] Landauer, R., “Irreversibility and heat generation in the computing process,” IBM J. Res. Dev. 5, 183-191, 1961.

[Malone 03] Robert C. Malone, John B. Drake, Philip W. Jones, Douglas A. Rotman, “High-End Computing in Climate Modeling,” contribution to SCaLeS report.

[Merkle 96] R. Merkel, E. Drexler, “Helical Logic,” Nanotechnology (1996) 325-339.

[MPI web] See Message Passing Interface (MPI) forum standards documents, http://www.mpi-forum.org/.

[NASA 99] R. T. Biedron, P. Mehrotra, M. L. Nelson, F. S. Preston, J. J. Rehder, J. L. Rogers, D. H. Rudy, J. Sobieski, and O. O. Storaasli, “Compute as Fast as the Engineers Can Think!” NASA/TM-1999-209715, available on Internet.

[NASA 02] NASA Goddard Space Flight Center, “Advanced Weather Prediction Technologies: NASA’s Contribution to the Operational Agencies,” available on Internet.

[Petrini 03] Fabrizio Petrini, Darren erbyson and Scott Pakin, “The Case of the Missing Supercomputer Performance, Achieving Optimal Performance on the 8,192 processors of ASCI Q,” in Proc. of IEEE/ACM SC2003, Phoenix, AZ, November 2003.

[Sair 03] Suleyman Sair, Timothy Sherwood, and Brad Calder, “A Decoupled Predictor-Directed Stream Prefetching Architecture,” IEEE Transations on Computers, Vol. 52, N0. 3, March 2003.

[SCaLeS 03] Workshop on the Science Case for Large-scale Simulation, June 24-25, proceedings on Internet a http://www.pnl.gov/scales/.

[Seitz 85] C.L. Seitz, A. Frey, S. Mattisson, S. Rabin, D. Speck, V. van de Snepscheut, “Hot-clock NMOS,” in Proc. of the 1985 Chapel Hill Conference on VLSI, 1985.

[Sterling 02] Thomas .L. Sterling and H.P. Zima. “Gilgamesh: A Multithreaded Processor-In-Memory Architecture for Petaflops Computing.” Supercomputing02, Baltimore, Maryland, November 18-22, 2002.

[Sunaga 96] Sunaga, T., Peter M. ogge, et al, "A Processor In Memory Chip for Massively Parallel Embedded Applicatiions," IEEE J. of Solid State Circuits, Oct. 1996, pp. 1556-1559.

[Timler 03] J. Timler an C.S. Lent, “Maxwell’s demon and quantum-dot cellular automata,” Journal of Applied Physics 94, 1050-1060 (2003).

[Tummala 99] Rao R. Tummala and Vijay . Madisetti, “System on Chip or System on Package?” in IEEE Design and Test of Computers, April 1999, Vol. 16 No. 2, pp. 48-56

[Upchurch 03] E. Upchurch, T. Sterling and J. Brockman. “Analysis and Modeling of Advanced PIM Architecture Design Tradeoffs,” in Proceedings of the 6th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA03), p66-77, 2003.

[Vieri 98] Carlin Vieri, M. Josephine Ammer, Amory Wakefield, Lars `Johnny' Svensson, William Athas, and Tom night, ``Designing reversible memory,'' in C. S. Calude, J. Casti, and M. J. Dinneen, eds., Unconventional Models of Computation, Springer, 1998, pp.386--405.

[Vieri 99] Vieri, Carlin, “Reversible Computer Engineering and Architecture,” Ph. D. Thesis, Massachusetts Institute of Technology 1999.

[Vitanyi 88] Vitanyi, P. M. B., “Locality, communications, and interconnect length in multicomputers,” SIAM J. on Computing, 17, 4 (1988), 659-672.

[von Neumann 56] von Neumann, J., “Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components,” in C. E. Shannon and J. McCarthy, Eds. Automata Studies. Princeton: Princeton University Press, pp. 43-98, 1956.