Review & Approval

Prepare Request
Search Requests
Generate Reports
Approvals
Help
Wizard

Search Requests
New Search
Refine Search
Search Results

Clone Request
Edit Request
Cancel Request

Search Detail

Submittal Details

Document Info

Title : Devices and Architecture for Maximum Supercomputer Performance
Document Number : 5221446 SAND Number : 2004-1545 C
Review Type : Electronic Status : Approved
Sandia Contact : DEBENEDICTIS,ERIK P. Submittal Type : Conference Paper
Requestor : DEBENEDICTIS,ERIK P. Submit Date : 04/16/2004

Author(s)

DEBENEDICTIS,ERIK P.

Event (Conference/Journal/Book) Info

Name : Supercomputing 2004
City : Pittsburge State : PA Country : USA
Start Date : 11/06/2004 End Date : 11/12/2004

Partnership Info

Partnership Involved : No
Partner Approval : Agreement Number :

Patent Info

Scientific or Technical in Content : Yes
Technical Advance : No TA Form Filed : No
SD Number :

Classification and Sensitivity Info

Title : Unclassified-Unlimited Abstract : Document : Unclassified-Unlimited
Additional Limited Release Info : None.
DUSA : None.

Routing Details

Role	Routed To	Approved By	Approval Date
Derivative Classifier Approver	YARRINGTON,PAUL	YARRINGTON,PAUL	04/16/2004
Conditions:			
Classification Approver	WILLIAMS,RONALD L.	WILLIAMS,RONALD L.	04/16/2004
Conditions:			
Manager Approver	PUNDIT,NEIL D.	PUNDIT,NEIL D.	04/16/2004
Conditions:			
Administrator Approver	ABEYTA,RAMONA D.	ABEYTA,RAMONA D.	12/11/2004
The conference paper needs the funding statement. See below. 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. 6/2/2004 (ra)			

Created by WebCo Problems? Contact CCHD: **by email** or **at 845-CCHD** (2243).

For Review and Approval process questions please contact the **Application Process Owner**

SAND2004-1545C

Devices and Architecture for Maximum Supercomputer Performance

Erik P. DeBenedictis, Sandia National Laboratories

***Abstract*—Supercomputer advocates optimistically predict performance will double every couple years, yet not every technology will survive. In this paper, we describe a path for supercomputing several decades into the future. This description includes a discussion of the physical limits, proposes new engineering methods, and gives an example of a supercomputer architecture that will be viable several decades from now. Since we write software with a lifetime of decades, it makes sense to understand which architectural trends of today will be around in several decades in order to run this software.**

We begin by exploring the types of supercomputer applications that stress supercomputers and clarify the need for higher performance in the future, concluding that there is a need for arbitrarily high performance.

We then study the future of the devices (currently transistors) that power supercomputers. We conclude that these will improve in accordance with Moore's Law for a 100× – 1000× performance increase over the next 1-2 decades, with a longer term trend to reach 10^8× - 10^{10}× today's performance levels.

We then explore a way of analyzing supercomputer architectures that lets us understand how close to the ultimate limits we can approach. With this method, we model an advanced architecture running a hydrodynamics application likely to be important and find that we can approach the theoretical limits to within an order of magnitude or so.

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

Supercomputers have become tools for science and defense and may become tools for national priorities and the progress of humanity.

Supercomputers today work on the principle of scalability. In this principle, a program is written to solve a problem of some given nature but of unspecified size. By sizing the computer to a problem, problems of most any size can be solved. There are many supercomputer programs in use today that can be scaled to the Petaflops or higher levels and solve problems of progressively more value.

For example, some populated regions are prone to Earthquakes. While the exact timing of an Earthquake is based on chaotic phenomenon and hence incomputable, it is possible to identify areas prone to violent shaking in the event of an Earthquake by creating a model for the geology under a region based on seismic data and then simulating statistically likely Earthquakes. There is seismic data today that if processed would identify safe and dangerous areas in Earthquake-prone regions (California, Japan, …) and thus save lives and property. Programs exist for this processing, but the required computing power is around 1 Exaflops, or 25,000 times the power of the most powerful computer today (the Earth Simulator) [Tromp 04, Donnellian 04].

As this paper is being written, the news media are carrying stories about concern that human-induced global warming will divert the Gulf Stream and cause Europe to enter a "deep freeze." It is plausible that the supercomputing community could be called upon to simulate the Earth's climate and decisively prove or disprove such a theory. To match the size of today's computers, today's climate simulation codes [Lin 04] use a cell size of several thousand cubic kilometers. If we scale down the cell size in these programs to 10×10×10 meters in order to capture known small-scale atmospheric weather effects (such as tornados), a climate prediction run would require around 10^{26} floating-point operations. This would require a computer of around 2 Zetaflops (Zeta is the SI Prefix for 10^{21}) running for a month, or about a 50 million times the power of the Earth Simulator. A computer of this power is beyond the reach of Moore's Law, although within the scope of proposed technologies.

The simulation of physics on a computer [Feynman 82] defines use of supercomputers today. This class of applications places specific demands on the architecture of the underlying computer. Figure 1 illustrates the division of space into a series of regions. Each cell holds the state of a particular

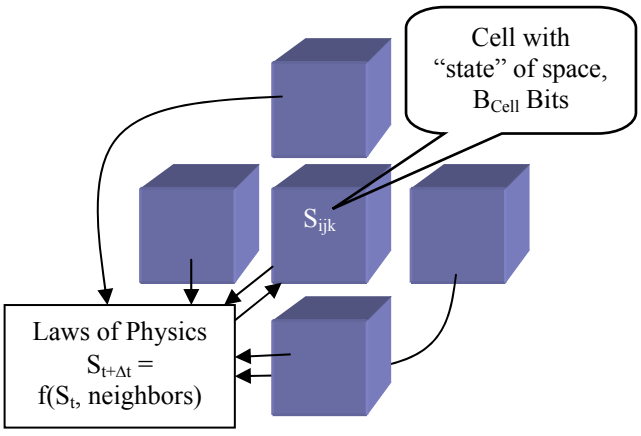

Figure 1. Simulation of Physics on a Computer. Each cell is comprised of B_{Cell} bits of computer memory representing the state of spatial area or volume in the problem. The computer updates the contents of each cell repeatedly for time intervals ΔT.

region of space, such as the composition of rock under a city or the composition and motion of air or water. The computer simulates the evolution of space by updating the state of each region for an interval ΔT, based on the computer evaluating the applicable laws of physics for all the regions.

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 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. Modern parallel computing systems do this with a function called Allreduce [MPI web], which is the name we will use in this paper. As illustrated in figure 2, a more complete description of the calculation involves processors evaluating the laws of physics on K cells, followed by a communications event across the entire machine.

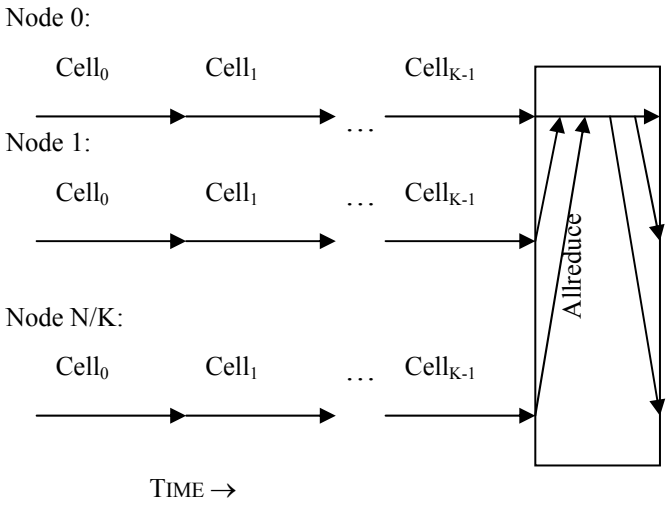

Figure 2. Organization of a Time Step. Each computer "node" evaluates the laws of physics for a group of K cells. Subsequently, all the nodes in the application synchronize and exchange data about ΔT and/or termination of the algorithm.

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 model for the geology under a city that matches seismic traces most closely. 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.

It is important to note that not every important problem has the characteristics described above: commercial computing is more I/O intensive and changes internal state much less; integer factorization for cryptology makes much more use of inter-processor communications.

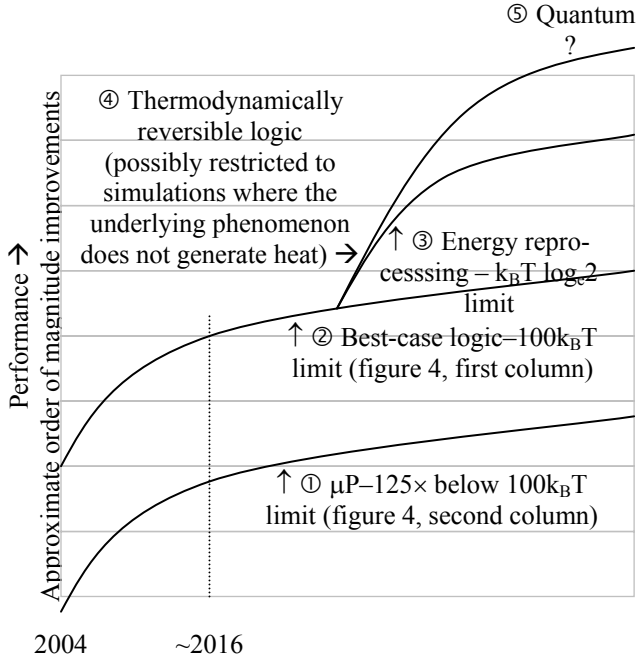

Figure 3. Projected technology sequence for supercomputer performance improvements.

II. The Limits of Computer Technology and the Scope of This Analysis

Figure 3 is the author's projection of a technology sequence that could apply to supercomputing. We are choosing to define supercomputing as the "simulation of physics on a computer" using algorithms formulated using floating-point operations consistent with figures 1 and 2. In considering technologies, we have rejected technologies that are not based on logic or floating point or would be slower, larger, or more power consumptive than reasonable extrapolations of today's computers. This eliminates many interesting technologies, such as neural nets, molecular computing, and biological computing.

Curve ② 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. As will be described later in this paper, the amount of energy must be greater than about 100 k_BT (where $k_B=1.38\times10^{-23}$ is Boltzmann's constant and T is the temperature in Kelvins) for the computer to be reliable. This mechanism is the predominate source of power usage in today's computers.

Notwithstanding the above, curve ① represents popular microprocessor-based supercomputers. Besides the floating-point units that move the user's program forward, the vast

majority of microprocessor logic is overhead for the management of the program itself. This logic includes instruction decoding, cache memory, memory busses, etc. Based on current architectures, curve ① has been drawn below curve ② by a factor of 125× to capture the efficiency loss in microprocessors.

Our view is that curves ① and ② represent the limits of a major generation of supercomputer technology, but not the end of progress. The multi-billion-dollar semiconductor industry has plans for virtually transparent improvements to underlying devices that will let us reach curves ① and ②. In contrast, curves ③ and ④ are supported by academic research into methods that are promising but not transparent.

The 1's and 0's needed to reach the higher curves are created as they are today, but their energy is recovered in curve ③ or recycled in curve ④ when no longer needed. Managing energy in this way substantially reduces the heat generated per operation and permits more useful work per watt, but it comes at a high cost.

Curve ③ represents a computer built of logic gates of the maximum possible efficiency. There is 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 two orders of magnitude below 100 k_BT. However, for a given technology, the number of devices in an energy-recovering gate is higher and the speed is lower than current designs. Thus, the cost of moving to curve ③ is that the number of devices (transistors) required per chip goes up substantially. Since it appears that the cost per device will continue to decline more or less indefinitely, this transition is very likely to occur at some point [Fredkin 82].

Recycling energy as required for curve ④ requires 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 are well-developed (albeit academic) studies of "reversible" logic gates that have the same numbers of inputs as outputs that do not destroy information. For example, a Fredkin gate with 3 inputs and 3 outputs simply exchanges two inputs in response to the third. Academics have found ways to construct computers with this alternate type of logic, but 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]. A full description of these methods is beyond the scope of this paper, but our point is that reaching curves ③ and ④ is likely to cause a disruptive change.

Finally, region ⑤ suggests that there may be technologies offering even more performance. Quantum computing offers the prospect of exponentially greater performance than any of the "classical" technologies on the graph. However, quantum computing is too new to be considered further in this paper.

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

Curves ① and ② are elaborated quantitatively in figure 4 for a computer the size of a mainstream ASCI supercomputer, such as the Sandia Red Storm system. Figure 4 has two analyses: a physics analysis going from the top down and a semiconductor industry projection going from the bottom up. The two meet in the center with a small but instructive gap.

Curves ① and ② share the $k_BT \log_e 2$ limit of irreversible logic. These limits of 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 yields the top number in figure 4 of 250 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, 12.5 ZFLOPS (Zetaflops, Zeta is the SI prefix for 10^{21}). This issue defines curve ③.

The impact of thermal noise on computer reliability will be the first effect to derail Moore's Law for supercomputer use. The reliability of computer logic was also explored by von Neumann [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

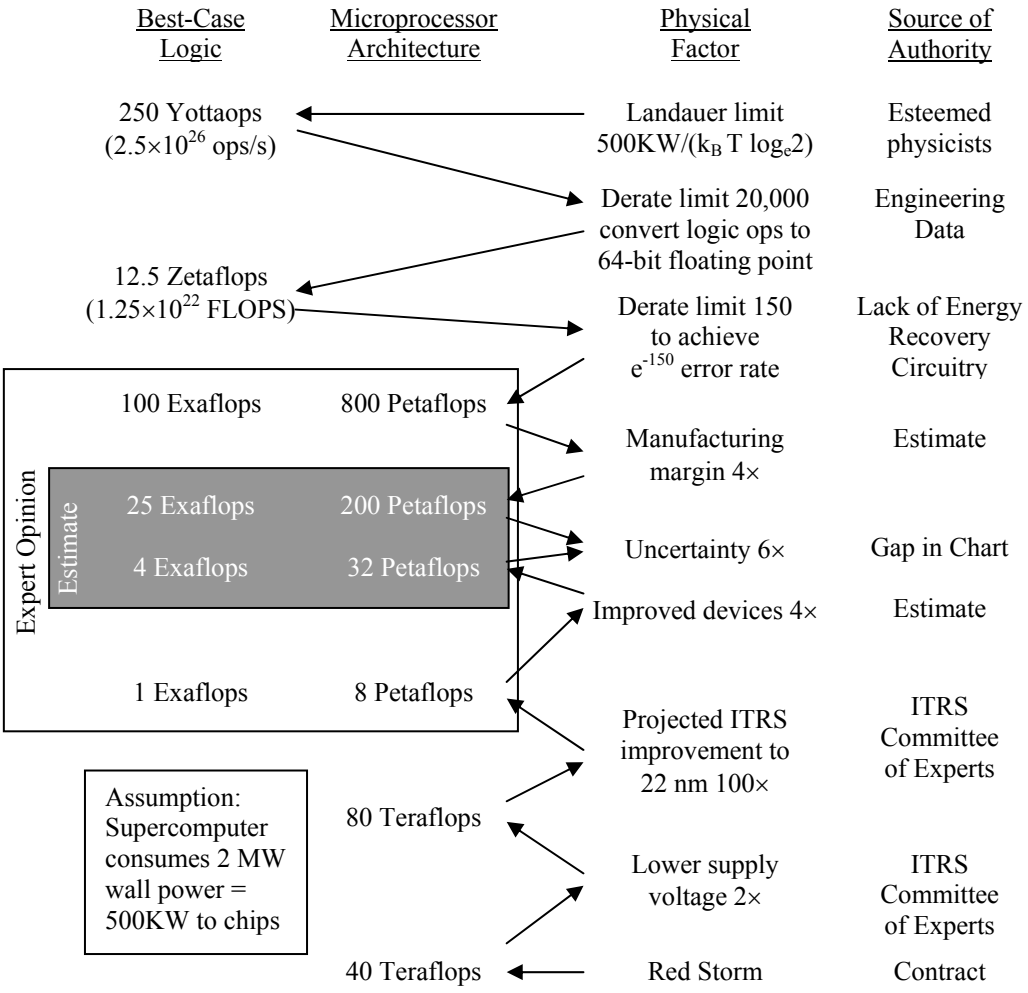

Figure 4. Limits on Supercomputers Set By The Laws of Physics. This chart derives the upper bounds on performance be derating the physical limits while simultaneously building up possible performance from known supercomputers and industry plans. A small region of uncertainty appears at the center.

exactly the same as the noise in FM radios (its magnitude is k_BT), 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 04] 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.

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

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	

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.

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 we have found [Frank 99] used a derating factor of 12×. However, this analysis was seeking an "expected value" rather than a "limit" and so we see it as 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 authors, 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. The authors' assertion is that it will not make a qualitative difference.

Figure 4 has direct applicability to the application examples presented earlier in this paper. To be specific, the Earthquake risk mitigation application is believed to require a computer of 1 Exaflops performance. From figure 4, this performance level is between the limits for a microprocessor based solution and a best-case logic one. This puts the Earthquake risk mitigation study forever out of the range of solution by Commodity Off The Shelf (COTS) clusters, unless the cluster is to be made substantially larger than the Red Storm system (>>2 MW power consumption).

The global climate problem described earlier and figure 4 provide an argument for moving beyond the reach of Moore's Law. The 2 ZFLOPS (2×10^{21} FLOPS) requirement described earlier exceeds the 100 EFLOPS peak in figure 4. To achieve this level would require energy reprocessing logic per figure 3 curves ③ or ④.

IV. Can We Reach The Limit?

We will provide a method for determining if it is possible to approach the limits of irreversible logic. We will then apply this method and show that we can approach curves ① and ② of figure 3; later, we will give a qualitative argument that we may be able to approach curves ③ or ④ as well. In this approach, the running time of an application is modeled on two hypothetical computers. The modeling builds on work in applications modeling [Christopher 04a, Christopher 04b, Hoisie 00, Kerbyson 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 authors 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

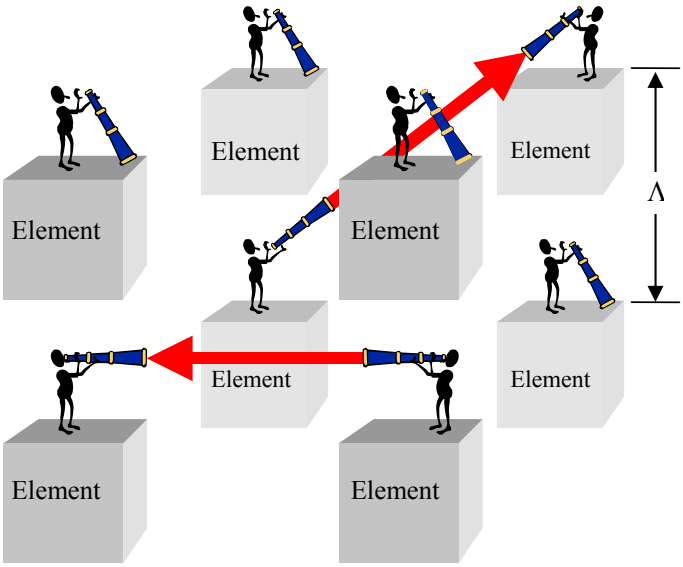

Figure 5. Aerogel Computer Model. Cells represent either gates of memory locations. Cells communicate through optical pulses that travel at the speed of light without interference.

of the volume of a nominal logic transistor or DRAM cell (we are 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.

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, we 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."

TABLE II
COOLING SYSTEMS

Method	C_x Capacity
Air	47 KW/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 KW/cm^2. [b]The theoretical limit for a system that is operated intermittently. Equivalent in calculations to an infinite cooling capacity.

V. AN EXAMPLE

We will develop the equation for applying the Aerogel model to an application as shown in figure 1 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 the most popular supercomputer application for the Department of Defense (DOD). The results of this analysis are plotted in figure 6.

The CTH program provides 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} =21Kbits/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 K cells = K × B_{Cell} bits worth of state, and some accessing logic as described below. We will also assume $\sqrt[3]{(N/K)}$ 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 K cells in size.

We describe elsewhere [DeBenedictis 04] 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 ~$\log_2 K$ 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. We also describe how to create a memory system that is quite energy efficient. 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 K cells plus the time for the global communications step, designated $T_{Allreduce}$ and described later:

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

The total number of elements in the supercomputer is given by

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

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 K evaluations than others.

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

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 at our 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 our 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 04].

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 K.

The program explores supercomputers running and SOR iteration on an N=n×n×n for n=50,000 cell space, where the cells are distributed evenly onto N/K 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 K to form the horizontal axis.

For each computer, the program optimizes the number of nodes per chip. While the number of nodes (each holding K 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 in available in [DeBenedictis 04], 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, K. The largest values of K correspond to a uniprocessor, with the entire problem in memory and a single CPU stepping through the cells one at a time. Values of K 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.

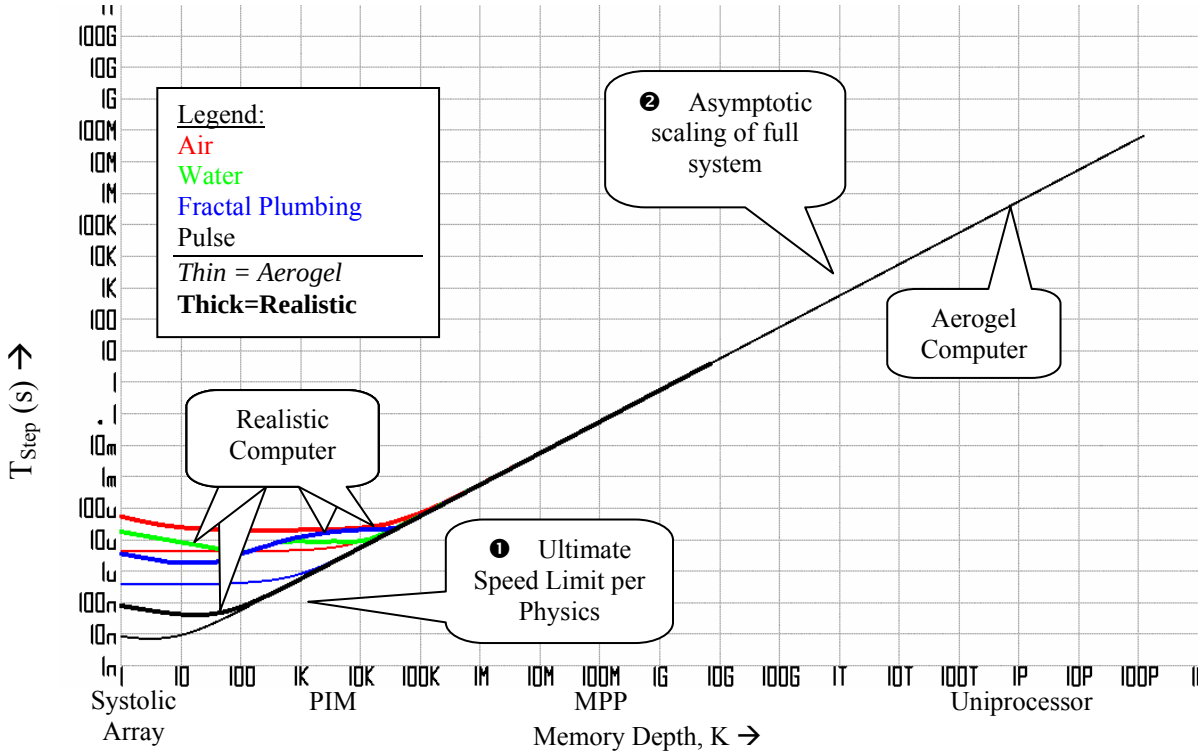

Figure 6. Results of Applications Modeling on Aerogel and Realistic Computer Models. Horizontal axis is the number of cells per "node," representing systolic arrays, PIMs, MPPs, and uniprocessors. The graphs show execution time per time step (lower is better) for various type of cooling technology. Thin 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.

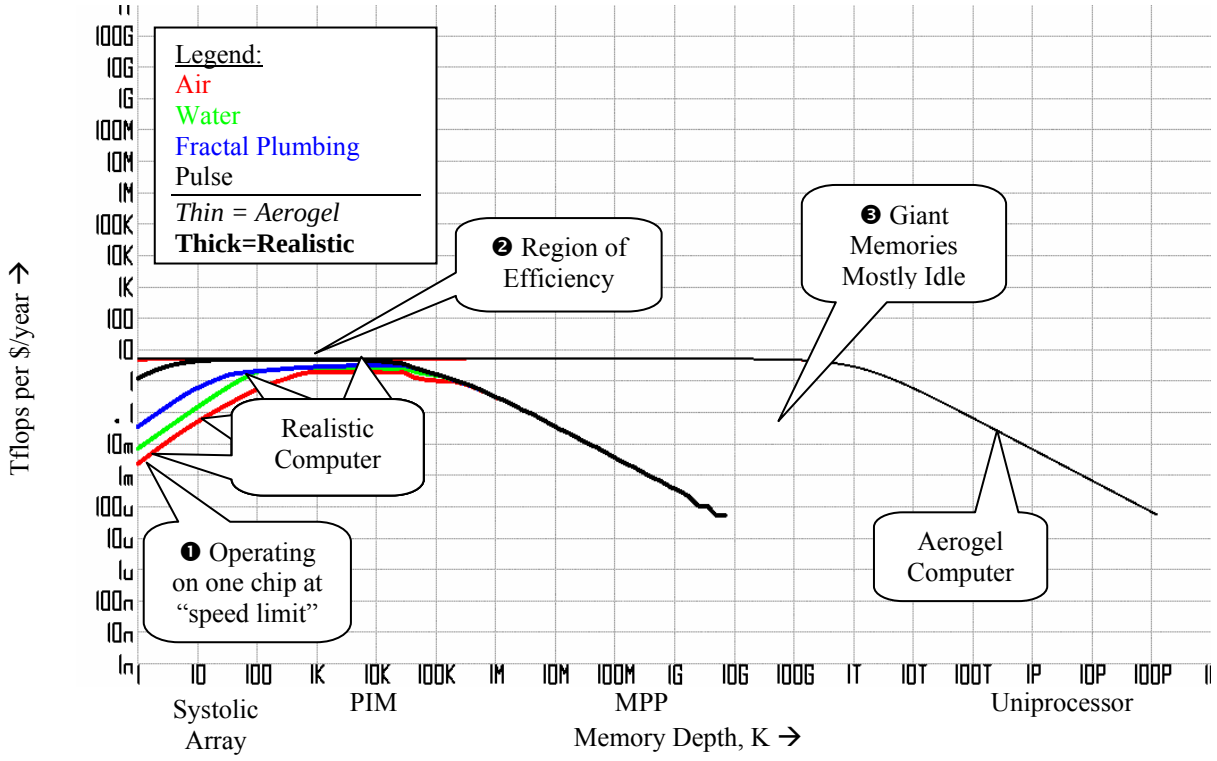

Figure 7. Cost Efficiency of Aerogel and Realistic Computer Models. Graph based on an economic model incorporating cost of equipment, electric power, and machine room space. Graph indicates broad peak of good efficiency.

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¢/KWH.

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 increased signal travel distance.

By performing this analysis, we claim to have shown that we have the technology to approach the theoretical limit of performance for an irreversible logic computer running the SOR algorithm. Through figure 4, we 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.

VI. What Breakthroughs Are Needed?

As an employee for a National Lab and thus somewhat associated with the US Government, the authors are 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 authors' 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 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.

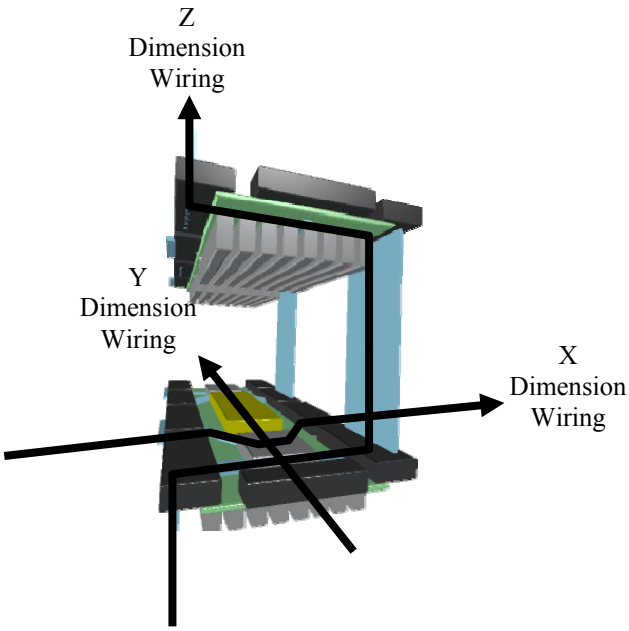

Figure 8A: Mapping of 3D Mesh to Physical Structure

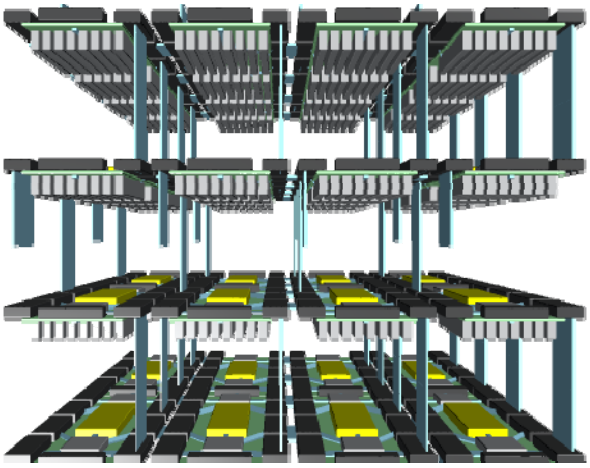

Figure 8B: Three Dimensional Packaging

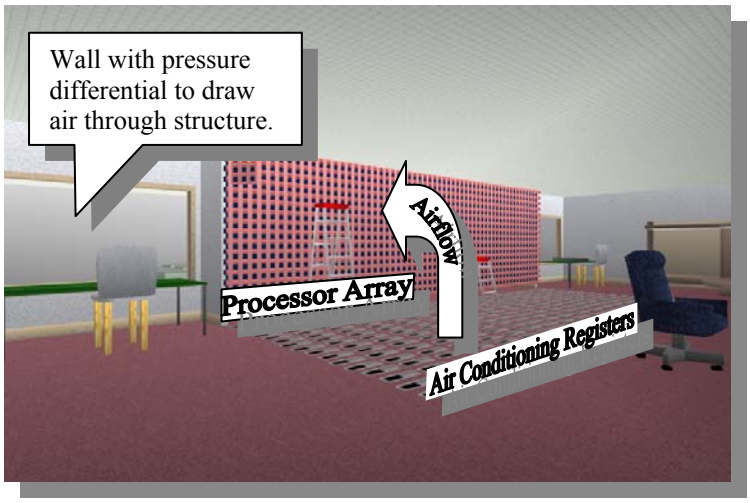

Figure 8C: Air-cooled configuration

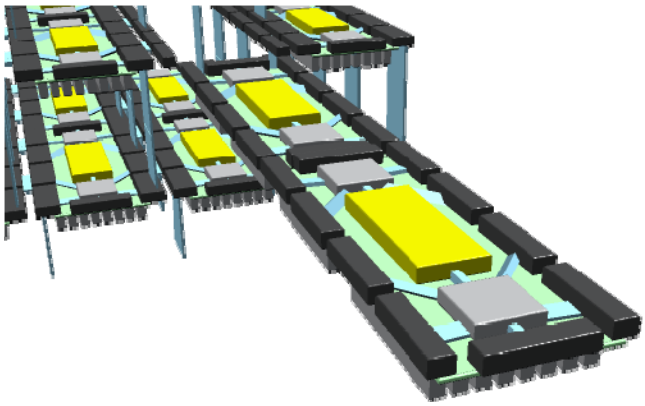

Figure 8D: Serviceability

VII. 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

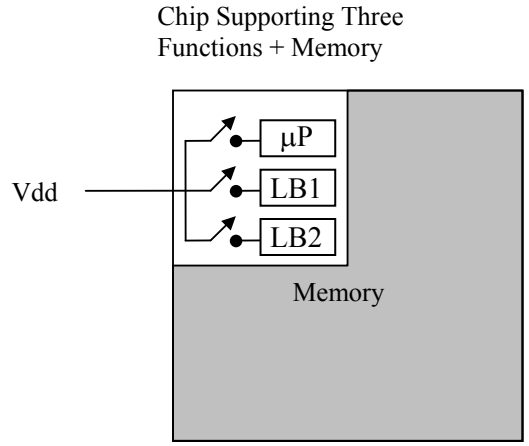

Figure 9. Multi-Architecture. Memory consumes 75% of chip area, but insignificant power. Each logic block consumes 75% of chip's power budget when turned on (only one on at a time) but insignificant area.

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.

VIII. Architecture of the Distant Future

It is not implausible that the architecture in figure 9 could persist for a very long time. Researchers in the field of "reversible logic" have created CMOS prototypes for thermodynamically reversible arithmetic [Kim 01], microprocessors [Vieri 99], memories [Vieri 98], and a "C" like programming language [Frank 97b]. These demonstrate the technology needed to follow curve ④ of figure 3, although taken at face value only for applications that do not destroy information.

However, researchers [Bennett 89] have shown a method of embedding general applications (i. e. applications that destroy information) within a larger framework that does not destroy information and can therefore be executed at the physical limits in accordance with curve ④. While these methods are not straightforward to apply in general and impose various overheads, they have been applied to "simulation of physics on a computer" as defined in this paper with a tractable overhead. Given that the timeframe for curve ④ is decades away, the author's position is that these methods are candidates for automatic translation (i. e. a compiler).

The considerations in the paragraphs above set the stage for the architecture in figure 9. Irrespective of the advanced technologies, a microprocessor following curve ④ will be less efficient than specially constructed logic. Thus, the approach in figure 9 of combining a flexible microprocessor with higher performance architectures can continue to apply to all the curves in figure 3.

There will be substantial challenges:

- We are aware of no long-term roadmap for packaging and inter-node communications. With reference to figure 8, we know of no technology that can communicate across a 3D structure at speeds approaching the speed of light, at sub k_BT energies per bit, and that can be disassembled for repair.
- The degree of parallelism is likely to be very high and this may cause programming problems. The speed of individual microprocessors or other processing units quickly becomes limited by the speed of light and performance increases come only through massive parallelism. While the parallel processing industry has been remarkably successful at scaling to 10,000 nodes, there is no assurance that this will be true at a million, billion, or trillion nodes.

IX. Conclusions

In writing this paper, our objective was to augment the belief that supercomputers will double in performance every couple years with a concrete path for making this happen. We identified how far into the future this trend can continue and identified places where major transitions in technology would be necessary.

Performance modeling for the Aerogel computer model was one of our key tools. The Aerogel computer has the property that no physically realizable computer can ever run faster than the optimal program running on a hypothetical Aerogel computer. If a realistic computer can come close (say within an order of magnitude) of the performance of an Aerogel computer, we can claim to be approaching optimal performance with a realistic design.

Our primary conclusion is that power will be the crucial factor in the future. The likely paths to increased performance will be:

- Architecture, by augmenting microprocessors that are about 1% power efficient with other approaches that are more efficient.
- Device improvements, including a factor of 100 as predicted by the semiconductor industry in the next dozen years and perhaps another factor of 10 beyond that.

Our projections for supercomputers reinforce the goals of some architectural initiatives already in progress by others. This refers to initiatives that provide more FLOPS per watt through tighter integration and architectural enhancements.

The optimum point in architecture seems to be nodes comprising a functional unit and a memory of approximately equal transistor counts. Given the inherent complexity of a floating-point unit, this comes to about a million transistors. With hundred billion transistor chips predicted in the next two decades, this suggests single chips with an unprecedented 100,000 processors.

The authors have found a way to run legacy code at microprocessor efficiency levels (1%) and new code at higher efficiencies. New code could be in libraries, enhanced instruction sets, or operating system. However, the authors have not found a way to make legacy code run at the highest rate without substantial rewriting.

The communications bandwidth needed by applications that "simulate physics on a computer" appears to be available without heroic measures. We find interchip communications

via wires connected to pins of today's density should be sufficient. However, we propose a 3D packaging scheme at the highest level. This design will offend people who prefer 19" rack mounted gear with connections running under the floor. Our analysis indicates that solution is not viable.

We anticipate further work in extending the analysis to applications other than CTH and device technologies that can reach curves ③ and ④ of figure 2.

REFERENCES

[1] [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.

[2] [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.

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

[4] [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.

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

[6] [Davis 04] Kei Davis, Adolfy Hoisie, Greg Johnson, Darren Kerbyson, 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.

[7] [DeBenedictis 04], Erik P. DeBenedictis, "Taking ASCI Supercomputing to the End Game," Sandia National Laboratories SAND report SAND2004-0959, March 2004.

[8] [Donnellian 04] Andrea Donnellan, John Rundle, John Ries, Geoffrey Fox, Marlon Pierce, Jay Parker, Robert Crippen, Eric DeJong, Ben Chao, Weijia Kuang, Dennis McLeod, Mitsuhiro Matu'ura, Jeremy Bloxham, "Illuminating Earth's Interior through Advanced Computing," IEEE Computing in Science and Engineering, Jan-Feb 04, 29-35 pp. 36-44.

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

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

[11] [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.

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

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

[14] [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.

[15] [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.

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

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

[18] [Kerbyson 01] Darren J. Kerbyson, 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.

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

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

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

[22] [Lin 04] Shian-Jiann Lin, Robert Atlas, and Kao-San Yeh, "Global Weather prediction and High-End Computing at NASA," IEEE Computing in Science and Engineering, Jan-Feb 04, 29-35.

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

[24] [Petrini 03] Fabrizio Petrini, Darren Kerbyson 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.

[25] [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.

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

[27] [Tromp 04] Jeroen Tromp, private communications, 2004.

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

[29] [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.

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

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

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

[33] [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.