"Empty Space" Computes:
The Evolution of an Unconventional Supercomputer

Jonathan W. Mills[1]   Bryce Himebaugh   Brian Kopecky

Matt Parker   Craig Shue   Chris Weilemann

Computer Science Department
School of Informatics
Indiana University
Bloomington, Indiana 47405
USA

ABSTRACT

Lee A. Rubel defined the extended analog computer to avoid the limitations of Shannon's general purpose analog computer. Partial differential equation solvers were a "quintessential" part of Rubel's theoretical machine. These components have been implemented with "empty space," or VLSI circuits without transistors, as well as conductive plastic. For the past decade research at Indiana University has explored the design and applications of extended analog computers. The machines have become increasingly sophisticated and flexible. The "empty" computational area is devoted to solving partial differential equations. The rest of the space includes fuzzy logic elements, configuration memory and input/output channels. This paper describes the theoretical definition, architecture and implementation of these unconventional computers. Two parallel applications are described in detail. Rubel's model can be viewed as an abstract specification for a distributed supercomputer. We close with a description of an inexpensive 64-node processor that was designed using our current single processor. The next step is to return to VLSI with an improved understanding of the architecture—and seek computation speeds approaching trillions of partial differential equations per second.

Categories and Subject Descriptors

C.1.3 [**Other Architecture Systems**]: Analog Computers, B.7.1 [**Types and Design Styles**]: Advanced Technologies, VLSI, C.5.1 [**Computer System Implementation**]: Super (very large) Computers, G.1.8 [**Partial Differential Equations**]: Multigrid and multilevel methods, F.1.2 [**Modes of Computation**]: Parallelism and concurrency, Probabilistic computation

General Terms

Design, Experimentation.

Keywords

Extended analog computer, general purpose analog computer, Lukasiewicz logic, hybrid digital-analog architecture

[1] Telephone: 1-812-855-6486 E-mail: jwmills@indiana.edu

1. INTRODUCTION

In 1995 the MOSIS educational service initially rejected a series of VLSI circuits because they consisted of 25 connections to "empty space," that is, silicon p-well, p-diffusion, n-well and n-diffusion without transistors. MOSIS staff asked if some of the layers in the design had been accidentally left out, but were told that the chips were indeed correct because "empty space" computes. This paper traces research at Indiana University over the past decade that has explored the architecture, implementation and applications of Rubel's extended analog computer, in which something is computed by implementing "nothing." Single processors have become increasingly flexible as applications have stretched their capabilities. Now, after inexpensive coprocessors have proved to be reliable, a prototype for a distributed extended analog supercomputer has been designed and is being built. A few nodes are already in use at other universities.

2. DEFINITION OF THE EXTENDED ANALOG COMPUTER

The extended analog computer (EAC) extends the general purpose analog computer (GPAC) as an abstract model of computation [Rubel, Shannon]. Rubel did not believe that the EAC could be constructed because it was too broad to be implemented with any single technology. Metal plates for heat diffusion, soap bubbles to model minimal surfaces, and vibrating strings and membranes to study the wave equation were three of the examples he presented. The EAC was described as an ideal paradigm that could not be implemented, just as a Turing machine is an ideal with an abstract definition that cannot be fully constructed, either. However, within limitations, each machine has been built. Turing machines are realized as digital computers with finite and bounded (not unbounded) memory and storage. Generalized EACs can be built to solve diffusion, surface minimization and the wave equation electrically, directly implementing diffusion with charge carriers, modeling surfaces with a $2\frac{1}{2}$D stack of processing elements, and using an external oscillator to generate harmonic "vibrations."

2.1 Computability and Measurability

Rubel defined the extended analog computer's operation in terms of arbitrary real numbers that were not required to be digitally computable, that is, generated by a Turing machine. However, during correspondence in 1994 with the primary author, Rubel restricted the scope of arbitrary real numbers in his search to prove limits on the computability of the EAC:

"Essentially, there is some [exotic] real number that does any preset job. I believe in analog computers that work with genuine real numbers, but not those that have [exotic] real numbers built into them. My philosophy is that the current notion of "function" is way too broad, and too far removed from intuition. This produces lots of pathological examples, and a badly distorted theory of functions. I am advocating a kind of return to the mind-set of Euler's time. But, as usual, I keep running into very hard concrete problems in mathematics, and it is notoriously slow, hard, and risky work to solve them. After about five years, I still can't prove that the EAC cannot generate ALL analytic functions" [Rubel].

Rubel introduced the term "genuine real," in part, because he was aware of work that proved how certain "exotic" real numbers were used to embed non-recursively enumerable languages under a Cantor set encoding. If these reals were permitted, and further, if even one such real could be reliably generated and measured to a finite but unbounded precision, then super-Turing computations would be physically possible [Siegelmann & Sontag]. One physicist suggested that this was not an issue for implementers of the EAC. In his words, "A real number measurable to tens of thousands of decimal digits would be perturbed by a grain of sand falling from another grain of sand on a beach on a planet circling Alpha Centauri" [Girvin]. This graphic example of the effects of even infinitesimal noise on Cantor-set-encoded analog computation was later formalized by a proof that in the presence of bounded noise no analog computation exceeds the computability of a Turing machine [Maass and Sontag].

All physically-realizable models of analog *and* digital computability appear to require some notion of instantaneous precision. In the Turing machine, the value in any cell on the tape is zero or one, for an instantaneous precision of one bit (although the tape may store very long, and thus precise, sequences of bits produced by the computation). In the EAC and other analog models of computation, the measured value of any setting, constant or output variable at any instant ranges from one bit to at most 24 bits. One approach to using a single output value as the result of the computation is to define a filter on a compact space centered on a measurable value [Blair 2005]. Many computations might be "hidden" in the space near this instantaneous value, but only one rational approximation would be accessible to the user via a finite measurement. This does not violate the uniqueness imposed by the extremely well-posed constraint of the EAC, but only points out that we cannot distinguish between *many* unique outputs. While the ideal EAC computes π exactly for an ideal user who can measure its value exactly, an actual EAC computes one of many approximations to π, for an actual user who can only measure its value approximately.

This does not prevent the EAC from generating large and precise numbers as its output, but the output would be a function of the duration of the computation, would depend on a sequence of instantaneous measurements at each level (each with a limited precision), and thus would not escape the limits of effective computation by demanding single precise measurements that are impossible to obtain in the presence of noise. These conditions indicate one path toward formally defining Rubel's "genuine real" number with real-valued metric spaces centered on rational (and measurable) numbers.

2.2 Hierarchical Structure

The extended analog computer is defined in terms of a hierarchy of levels (N, N+^{1}/$_2$), (N+1, N+1+^{1}/$_2$), (N+2, N+2+^{1}/$_2$), … with each level composed of elements that may be components, some of which are "black boxes" representing instances of mathematical operators, such as set projection, or principles, such as analytic continuation. The hierarchy and the elements are finite but unbounded, and "connected with a great deal of feedback" [Rubel]. One consequence of so much feedback is that the EAC is naturally a dynamical system (although we do not yet completely understand how to design applications to exploit this property). Computations at higher levels are more versatile, a natural result of increasing computational complexity. All outputs are differentially algebraic, which for those results that are produced by ordinary differential equations (and analogously for partial differential equations) requires the functions to be differentiable at each level, or C^{ω}.

2.3 Elements

A review of the general purpose analog computer is presented to set the context for the extended analog computer.

2.3.1 General Purpose Analog Computer

Rubel stated that the general purpose analog computer "is really a mathematical concept" [Rubel 2]. It has only four kinds of "black boxes" that are "hooked up with lots of feedback" like the extended analog computer, but only a single level of hierarchy, unlike the EAC. Inputs and outputs of the GPAC were assumed to be real numbers, but no mention was made about their arbitrariness or computability by a Turing machine. The GPAC has long been considered to be less of a "general purpose" machine than a digital computer because of its lack of precision when implemented. However, it was recently shown that a Turing machine can be defined in terms of two GPAC integrators for each cell of the Turing machine tape [Graca 2005]. Unfortunately, such an implementation would be inefficient because so many integrators, each an operational amplifier, would be needed for even a small computation.

The following are the elements of a GPAC, which are realized with "black boxes" and other components:

Initial setting and constants. Initial settings and constants were not as carefully distinguished in the GPAC as they were later in the EAC. For example, constants of integration, *C,* were referred to as the "initial settings" of an integrator.

Independent variables. Outputs of the "black boxes" are the independent variables of the GPAC. Rubel stated that "any voltage that can be read in the circuit" is called an "output." This is the closest mention of the need to measure values, but does not reach the notion of instantaneous precision.

Hook-ups and feedback (wires). There are limitations on the interconnection of the "black boxes" similar to those of the EAC (for example, no two outputs connected to the same input) given in a paper by Pour-El [Pour-El].

Adders. An adder sums two values *u(t)* and *v(t)*, which are functions of time, to produce the sum *u+v* at *t*. The operation of the GPAC as a function of time (not space) is so pervasive that the index *t* is left out of the remaining definitions. The

concatenation operation over some whole number of adders generates the $\sum$ operation.

Multipliers. For inputs *u* and *v* a multiplier produces *u · v*. The *concatenation* operation over some whole number of multipliers generates the $\prod$ operation.

Integrators. For two inputs u and v, an integrator produces the output $\int_0^t u(s)dv(s) + C$. The *concatenation* operation over some whole number of multipliers generates multiple integration, $\int_0 \int_1 \int_2 \dots \int_k$. Because integration is defined as a function of time, *t*, and not space, concatenation does not yield integration over an area or volume in *n* dimensions. Rubel added differentiators and the "boundary-value problem" box to address this limitation.

The next section shows how the idea of machine-as-mathematical-concept influenced the definition of the EAC.

2.3.2 Extended Analog Computer

The extended analog computer has a more extensive set of "black boxes" and components than the general purpose analog computer. Certain operators left undefined in the GPAC, such as analytic continuation, were made explicit in the EAC. Other operations, such as substitution and inversion, were included to support functions that the GPAC could not compute. As Campagnolo noted, these additions may extend the computability of an EAC beyond Graca's construction of a Turing machine with a GPAC. Rubel wrote, "It is an unsolved problem whether [the EAC] can produce *every* real-analytic function. If it could, the EAC would be too broad to be interesting" [Rubel].

These are the elements of an EAC, which are realized with "black boxes" and other components. Those elements that are implicit in properties of matter, or semantic attribution, are noted in the definition. Elements that implemented physically are described in Section 2.5 Explicit Components.

Initial setting and constants. Initial settings s_1, s_2, s_3, … and constants c_1, c_2, c_3,… are fixed, arbitrary real numbers that are not required to be rational or digitally computable, that is, able to be generated by a Turing machine.. The only distinction between them is that initial settings are only produced at the first level, N=0, in the machine, while constants may be produced at any level.

Independent variables. These variables, x_1, x_2, x_3, …, are arbitrary real numbers produced at any level N of the EAC. Rubel stated no explicit requirement that they be measurable.

Hook-ups and feedback (wires). The initial settings, constants, independent variables, and the inputs u_1, u_2, u_3,… and outputs v_1, v_2, v_3,… of all the varieties of "black boxes" are connected by wires defined by pairs (w_i, w_j) from the Cartesian product W = $(s_1, s_2, s_3, \dots \times c_1, c_2, c_3, \dots \times x_1, x_2, x_3, \dots \times u_1, u_2, u_3, \dots \times v_1, v_2, v_3, \dots)$. There are three constraints: (1) the outputs of any level N can only be used as inputs at level N+1/2, N+1 and higher, (2) no two outputs can be connected to the same input, and (3) each input must be connected to at least one output.

There is a problem with this definition. Rubel was vague about the topology of interconnections, but stated that there was "a lot of feedback." Yet, according to this definition,

feedback is only possible locally within a half-level N or N+1/2, but not more widely. Our implementations have not followed this restriction in practice, akthough in general the machine operates without recursion as computation progresses "upward" from one level to the next.

Adders. Adders sum the vectors $u_1(x_1, x_2, x_3, \dots x_k)$ and $u_2(x_1, x_2, x_3, \dots x_k)$ to yield $u_1(x_1, x_2, x_3, \dots x_k) + u_2(x_1, x_2, x_3, \dots x_k)$ This operation is similar to the concatenated adders in a general purpose analog computer. Adders are implicitly implemented in the conductive sheets and the fuzzy logic units, with operation governed by Kirchhoff's Current Law (itself based on the Law of Conservation of Energy).

Multipliers. Multipliers input the vectors $u_1(x_1, x_2, x_3, \dots x_k)$ and $u_2(x_1, x_2, x_3, \dots x_k)$ to yield $u_1(x_1, x_2, x_3, \dots x_k) \cdot u_2(x_1, x_2, x_3, \dots x_k)$. This is similar to the concatenated multipliers in a general purpose analog computer. Multipliers are implicit operations of the conductive sheet (scaling by a resistive constant) and the fuzzy logic functions (slope of a curve).

Substituters. For a vector of values $v(x_1, x_2, x_3, \dots x_l)$ and the input vector $u_1(x_1, x_2, x_3, \dots x_k)$, …, $u_l(x_1, x_2, x_3, \dots x_k)$ the EAC replaces each x_1 in $v(x_1, x_2, x_3, \dots x_l)$ with the corresponding value $u_i(x_1, x_2, x_3, \dots x_k)$ to yield $v(u_1(x_1, x_2, x_3, \dots x_k), \dots, u_l(x_1, x_2, x_3, \dots x_k))$. Substituters are implicit functions of the connection of one component, such as a conductive sheet, to another, introducing value(s) to be substituted from the outputs of other conductive sheets or fuzzy logic functions.

Inverters. For a well-defined C^ω function, the inverter "locks down" the outputs and generates the inverse of the function, yielding the inputs $u_1(x_1, x_2, x_3, \dots x_k)$, …, $u_l(x_1, x_2, x_3, \dots x_k)$ that yield $f(u_1(x_1, x_2, x_3, \dots x_k), \dots, u_l(x_1, x_2, x_3, \dots x_k))$. Inverters are semantic attributions of a level of the EAC, which solves the inverse of a function, for example, by implementing backpropagation in a neural network as used to implement a character recognizer [Mills 96].

Differentiators. For $f(x_1, x_2, x_3, \dots x_k)$ a differentiator outputs a possibly mixed partial derivative $Df(x_1, x_2, x_3, \dots x_k)$

$$ Df = \frac{\partial^{\alpha_1 + \alpha_2 + \alpha_3 + \dots + \alpha_n} f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \partial x_3^{\alpha_3} \dots \partial x_n^{\alpha_n})} \quad . $$

Holding a variable x_i fixed is implemented by forcing it to a constant *k*, or, over a series of points in space, a function of the variable $f(x_i)$ such that it is not influenced by the partial derivatives of the other variables. Differentiators are implicit in the conductive sheets. Simpler versions are semantic attributions of Lukasiewicz logic elements, which model Laplacian differentiators, well-known as edge detectors [Laplace].

Set theoretic operators: >0, ≥0, union, intersection, projection. The set theoretic operators provide comparison and combinations of functions of variables $f(x_1, x_2, x_3, \dots x_k)$. It should be noted that in finite time it is not possible to exactly compare any value to zero, because the sequence of digits in a real number such that its partial representation is 0.00000000… may have some digit beyond those checked that is non-zero. This is not an issue for a theoretical model,

but is an example of the limits of measurability for a physical implementation. These operators are implicit in the connections between sheets, which are passed through the fuzzy logic functions. An example of the comparisons that occur between sheets is found in a radiosity-based image rendering application designed by Ololoweye, where the intensity of light absorbed by an illuminated object was "cut off" by a fuzzy logic element before being passed to the display (the two dark squares in Figure 1) [Bayo].
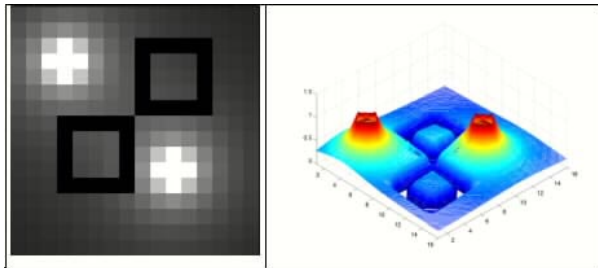

Figure 1. Set theoretic comparison X>0

"Boundary-value problem" box. This is the "quintessential black box," according to Rubel, and our work has supported this. It directly solves a system of partial differential equations, including some ordinary differential equations, subject to some boundary conditions. In the architecture of the EAC this element is explicitly implemented with conductive sheets, known since the 1950's to solve Laplacian and Poisson PDEs using materials such as carbon paper or resistive films [Karplus]. Silicon and conductive plastics continue this historical technique using modern materials.

Restricted limits. The restriction on taking limits is enforced by only permitting boundary values that have been computed at the immediately prior level N–1 in the EAC. Permitting an unbounded series of boundary value computations would permit the EAC to compute *all C^ω-functions*. Then, as Rubel wrote, "we would have no "computer" at all" [Rubel]. In implementation this is a constraint on the physical connections between components—and one we ignore, as there are so few levels, even in the prototype supercomputer (no more than 64 at present), as to take "unrestricted" limits.

Analytic continuations. The analytic continuation "black box" is a mathematical property of a function such that one point defines other points with a neighborhood. Practically, this is a condition of interpolation, and barring discontinuities in the material from which the EAC is fabricated, is implicit in the regularity of matter at the macroscopic, classical level.

Extremely well-posed determinism. This form of determinism enforces a compact space on the output at any point in the computation. This does not say that there cannot be sharp gradients or rapid changes in a function, but it does demand that any perturbation by some small amount ε produces a change in the resulting output that is a C^ω-function $f(\varepsilon)$. Extremely well-posedness is also enforced by the Law of Conservation of Energy, which prevents gross discontinuities in the output of the EAC. For example, taking the derivative of a 0-to-1 step function, such as occurs at the edge of an image, is theoretically infinite but in practice is limited by the available power in the circuit, This has been observed in Lukasiewicz logic arrays acting as a Laplacian differentiators.

2.4 Architecture

The extended analog computer is a restricted form of direct implementation architecture, generalizing an idea proposed for digital computers [Hoevel and Flynn]. As used here, it defines an EAC composed of one or more levels, each containing one partial differential equation (PDE) solver, some fuzzy logic functions, their internal and external interconnections, and a variety of input and output interfaces. The general architecture is shown below (Figure 2).
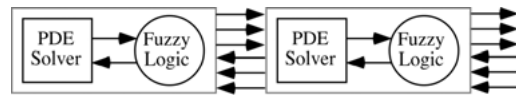

Figure 2. EAC general architecture

Only five elements are explicitly implemented in the architecture. These "carry" the rest of the EAC's elements, which are not physically distinguishable but are implicitly embodied in the physical properties and semantic attributions of "empty space" (that is, the conductive surfaces or solids), the fuzzy logic units, and the wires that connect all components. Even the wires compute in an EAC.

Digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are included to interface the EAC to digital systems, sensors and networks, including the worldwide web. Even the earliest VLSI implementations included digital registers to configure the machines. For flexibility, the current version of the machine uses discrete emulations of the fuzzy piecewise linear functions. However, these could be implemented with Lukasiewicz logic arrays, as they were in earlier versions.

2.5 Explicit Components

Initial setting and constants. Initial settings and constants are input points to the EAC. In the most recent version they are produced by the output of DACs precise to 10 bits that control current sources of slightly less precision due to noise that are connected to the conductive sheet and the fuzzy logic units.

Independent variables. These variables correspond to measurement points in the EAC. In the current version of the EAC these may be voltages measured directly by ADCs precise to 12 bits, or currents computed by Ohm's Law that are converted to digital values and sent back to an input DAC.

Hook-up and feedback wires need no further description. In the most recent version of the EAC the topology of the connections is configurable.

Conductive sheets and solids have been mentioned, and can be implemented easily. Any conducting or semiconducting material that can be formed into a sheet, layers of sheets, or a solid, and to which connections can be attached as ohmic point, line or area contacts will work. Silicon VLSI ("empty space"), conductive plastic foam, and gelatin "doped" with table salt have all been used in our machines.

Lukasiewicz logic arrays (fuzzy logic units). Lukasiewicz logic is used to implement the fuzzy logic units because of a theorem by McNaughton, which proves that sentences in Lukasiewicz logic approximate algebraic differential equations (ADEs) arbitrarily closely [McNaughton]. Because the general purpose analog computer computes ADEs, this correspondence implicitly includes the functionality of the elements of a GPAC.

3. IMPLEMENTATIONS OF THE EXTENDED ANALOG COMPUTER

The implementation of the extended analog computer is simple. The architecture is mapped to a conductive sheet, one or more arrays of fuzzy logic function units, current sources, current sinks, inputs and outputs, all connected by a reconfigurable array of wires. Input is obtained from potentiometers, digital-to-analog converters, or sensors (which may also be computing elements as is the case in the silicon VLSI single-pixel retina [Mills 1996]). Output is measured directly, or through analog-to-digital converters.

3.1 Breadboarded "Foamputer"

The "foamputer," a primitive extended analog computer, implemented one level of the EAC. The photograph shows the first prototype ever built (Figure 3).
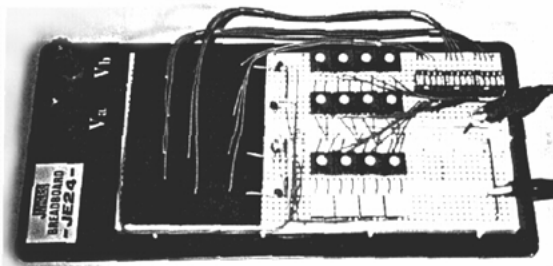
Figure 3. First EAC (1995)

On the left is a PDE solver cut from conductive plastic foam. Inputs were generated by twelve potentiometers on the right, that produced variable values for each "frame" of a computation, although some were treated as constants. Outputs were measured on the foam with a volt-ohmmeter. The same general structure with greatly improved flexibility and functionality is used today.

3.2 VLSI Circuits

A photomicrograph of a VLSI EAC built in 1996 illustrates its similarity to the general architecture (Figure 4).
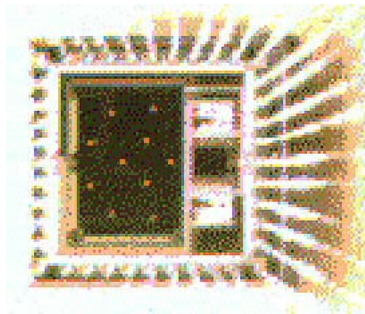
Figure 4. VLSI EAC (1996)

This circuit is still in use ten years later, having proven resistant to humidity, transient electrical shocks and dust, needing only to be cleaned occasionally with a soft brush. The VLSI sheet is enclosed in a lateral "ring" diode to prevent migration of charge carriers across the boundary of the sheet, "sharpening" the gradient manifold generated by the conductive surface (Figure 5).
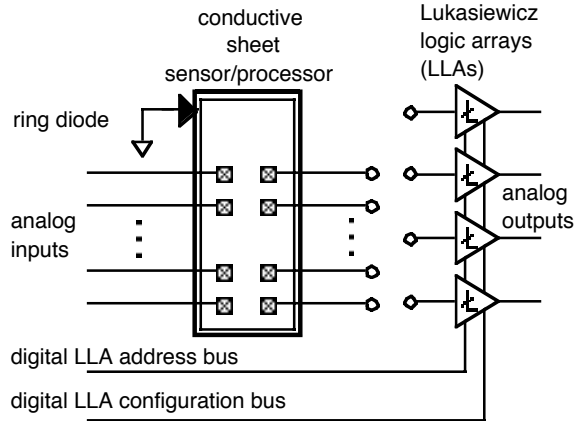

Figure 5. VLSI EAC block diagram

3.3 Unconventional Non-Silicon Designs

Several factors led to the design and implementation of unconventional non-silicon extended analog computers. The VLSI circuits, although digitally reconfigurable, needed complex interface circuitry. These chips were also difficult to use because their design had been "frozen" too early, before we understood the paradigm they represented. More time was spent trying to operate the circuits than exploring applications for them. By 1998, we had lost the departmental staff and expertise that supported the fabrication of VLSI circuits (under state law, Indiana University does not have a school of engineering, or an electrical engineering department).

This turned out to be fortunate. Several years were spent modeling EACs with MatLab and Mathematica®. The understanding gained from this exercise led to an improved series of EAC designs. We returned to the material used in the first prototype, conductive plastic foam, which protects digital integrated circuits from electrical shocks during shipment. It is cheap, readily available, and can be "fabricated" for use in minutes with a pair of scissors. Unconventional computers were built out of necessity, yet by choice. Plastic and discrete components permitted a cycle of rapid prototyping: design-application-redesign. This cycle has continued for over five years, and led to the EACs described in the next sections.

3.3.1 Networked Extended Analog Computer

The design of the VLSI extended analog computer was extended by adding additional fuzzy logic units (Lukasiewicz Logic Arrays, or LLAs) and programmable current sources and current sinks. Digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) were added to automate readouts and interface the EAC to digital computers and networks, including the worldwide web. Some memory was also provided to configure the machines digitally.

One of the four Internet-accessible EACs is shown with its components labeled (Figure 6). The analog (continuous-valued) Lukasiewicz logic arrays are digitally configurable (A). An Ethernet interface connects the EAC to the worldwide web (B). Analog-to-digital converters translate outputs to digital values for transmission and measurement (C).
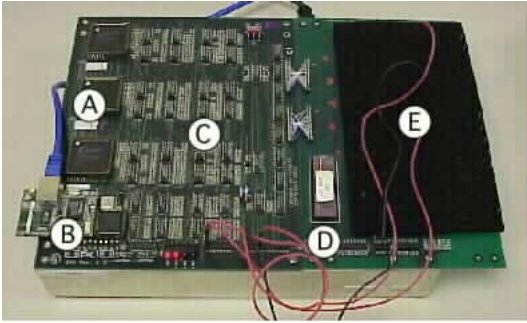
Figure 6. Networked EAC (2004)

A socket permits different components to be used (D). A VLSI chip is shown here, but Jell-O® brand gelatin was also connected to the EAC to study three-dimensional colloidal computers (Figure 8). Cultured neural tissue, organic semiconductors, and other materials can be used instead. The prototype is configured with a sheet of conductive foam that solves partial differential equations in microseconds (E). The VLSI chip at (D) solves the same PDEs in nanoseconds. The schematic is shown below, labeled accordingly (Figure 7).
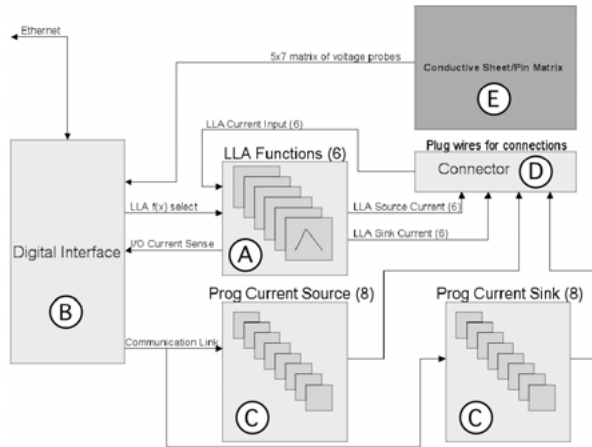
Figure 7. Networked EAC block diagram

3.3.2 Prototype 3D Extended Analog Computer
A prototype 3D extended analog computer was built with unflavored Jell-O® brand gelatin (Figure 8) [Miller].

Figure 8. Prototype 3D EAC (2005)

Sodium chloride (table salt) was added to the mixture before use. The resulting conductive gel had a 3×3×3 grid of electrodes on non-conductive plastic rods molded into it. Its operation was more accurate than the 2D foam. Several experiments were run with heterogeneous gels, adding plastic chips or short lengths of stripped wire. Their ability to model systems such as extraction of oil from shale is promising.

Conductive plastics and semiconductors can be injection molded or formed in layers with electrodes embedded or sandwiched between them. Further research into 3D EAC design is supported by the success of this prototype.

3.3.3 USB Networked Extended Analog Computer
The networked extended analog computer designed in 2004 used connections to the conductive sheet and VLSI circuit that were placed manually. This was soon found to be a severe limitation when students began to evolve EAC configurations with genetic algorithms (GAs). Even though the configuration of the EAC matched the genome closely, and efficient configurations were obtained in as few as a few thousand trials in ten or fewer generations, the results were limited to topological structures manually placed in advance.

A fully digitally-configurable EAC that was compact, low power, and portable (so students could take it home for research) was designed in response [Himebaugh] (Figure 9).

Figure 9. USB networked EAC (2006)

Every element that can be configured, from the selection of input and output contacts, values of current sources and sinks, even the shape of the emulated fuzzy logic functions, can be evolved under the control of a digital host computer with a USB port. The schematic is shown below (Figure 10).
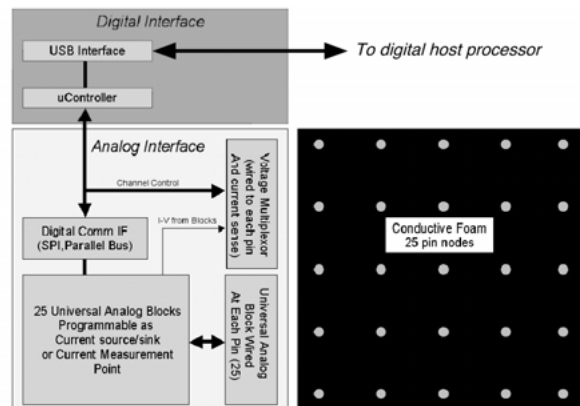

Figure 10. USB networked EAC block diagram

4. APPLICATIONS

The extended analog computer operates by analogy. Two systems are brought into congruence with each other. One system is the problem to be solved or, more generally, an application—a set of similar problems to be solved on a broad range of data sets—and the second is the EAC. This agreement between systems is first devised by visualizing it, then sending instructions to the EAC to configure it. These instructions resemble an assembly language program for a digital computer. This mode of configuring an EAC is just as tedious as is assembly language programming for a digital computer. Two research projects address this issue.

4.1 Visual Development Environment

Research is underway to devise a visual configuration interface that will permit the user to interactively "sculpt" an exemplar configuration using force-reflective gloves and stereo eyeglasses in an interactive three-dimensional environment, then observe its application to a large and varied data set. The first primitive example of this "Virtual Light" interface, implemented using Indiana University's computer-assisted virtual environment (CAVE), lets the user adjust the parameters of the EAC as it simulates a tissue-level neural network model of exclusive-OR (Figure 11) [Williams].
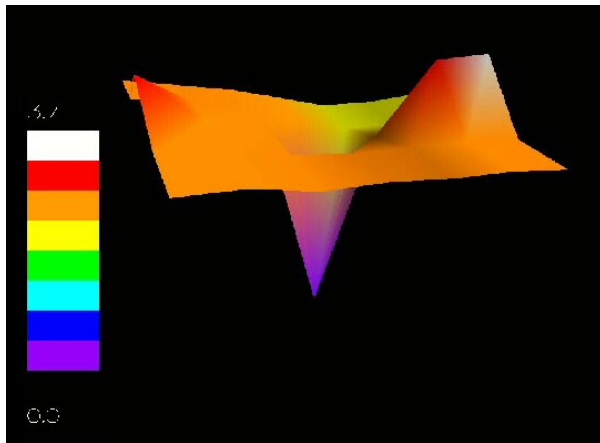

Figure 11. "Virtual Light" development environment

4.2 Compiler for Semantic Specialization

The semantic gap between the physical system and the EAC is small. In our experience, it is often easier to devise a visual model that describes some physical system and then map the application directly to the EAC than it is to translate the system of partial differential equations alone into a configuration. However, most potential users immediately ask how to automatically compile a system of PDEs directly to the EAC. At present this is poorly understood.

One approach to the problem is to examine how one "thinks up" the configuring analogy. Butterfly wing pattern generation illustrates the mapping of an analogy and its associated partial differential equations for reaction–diffusion to an EAC. This operation is "semantic specialization," or the assignment of meaning to the EAC. Examining the equations immediately shows the relationship between the partial derivatives and the conductive sheets. Coefficients match fuzzy logic functions, but less clearly. However, the topology

of the system is not present in the equation. Thus some form of visual or spatial annotation is necessary (Figure 12).
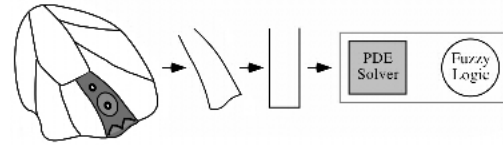
$$\frac{\delta a}{\delta t} = \frac{ca^2}{i} - k_1 a + D_a\left(\frac{\delta^2 a}{\delta x^2} + \frac{\delta^2 a}{\delta y^2}\right)$$

$$\frac{\delta i}{\delta t} = ca^2 - k_2 i + D_i\left(\frac{\delta^2 i}{\delta x^2} + \frac{\delta^2 i}{\delta y^2}\right)$$

a	concentration of activator
i	concentration of inhibitor
c, k_1, k_2	reaction constants
D_a, D_i	diffusion coefficients

Figure 12. Semantic specialization

This approach is supported by the experience of students who are introduced for the first time to the EAC. While direct semantic specialization, or thinking *analogically* instead of *algorithmically,* is initially an unfamiliar mode of thought for them, they eventually use *both* ways of thinking to develop applications. The students' first exercise is to generate the letters from the "butterfly alphabet" (Figure 12) [Sandved].

Figure 13. The "butterfly alphabet"

Nijhout's source-sink model of butterfly wing pattern formation was used, rather than a trivial "dot-matrix" solution (Figure 14) [Nijhout]. As a result students learned to devise an analogy for the spatial system, then apply their experience creating algorithms to write the configuration "program." This process could be performed by a compiler designed for semantic specialization, and is the subject of current study.
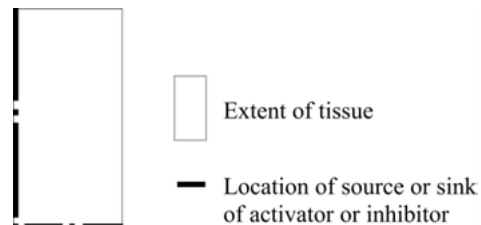

Figure 14. Nijhout's source-sink model

4.2 Pattern Generation and Image Recognition

Image recognition is used to maneuver robots and unmanned vehicles, to identify faces for security purposes, and one day may identify objects in images for data mining. Digital computers match patterns with complex serial computations. However, an extended analog computer can analyze all parts of the image at one moment, "seeing" it as a whole image rather than as several edges and areas. Previously, silicon "retinas" performed image recognition subtasks, such as edge-detection, corner recognition, and motion detection [1, 2, 3]. A single-sensor continuous retina used a semiconducting photosensitive silicon sheet to merge these tasks, and as an example of its function, was used to differentiate between two letters of the alphabet [Mills]. That experiment was limited because it used a 3x3 array of sampling points on the sensor, summed their outputs by wires, and learned to distinguish letters by selecting from only 27 fuzzy logic functions. We recently evolved letter recognition for all 26 letters of the Roman alphabet, extending that work substantially.

4.2.1 Approach

Instead of devising a setup of lights and lenses to focus an image on the silicon chip, a photosensitive array was built and attached to the screen of a computer monitor. Its output was sent to the conductive sheet on a networked EAC (Figure 15).
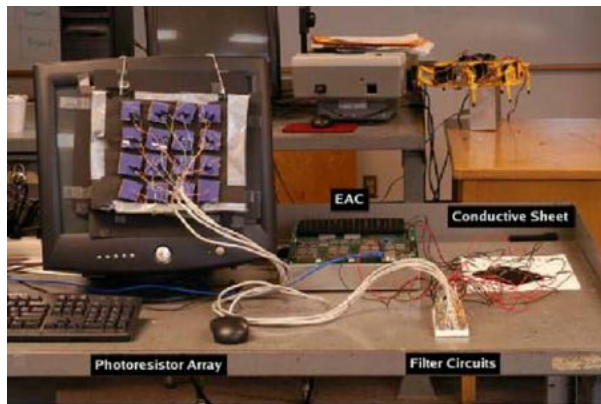

Figure 15. EAC as evolvable image recognizer

The evolved linear functions emulated customizable LLA functions digitally. Customizable LLA functions have been implemented in the USB networked EACs (Section), but they were not available at the time this work was performed.

Training was evolutionary, using a genetic algorithm to represent the form of each of four LLA functions. After a letter appeared on the screen and the voltages were output to the sheet, each of the four current readings from the conductive sheet went through their own digital LLA. The chromosome for each individual letter was made up of 128 genes of 6 bits each. Each LLA was given 32 genes.

A program automatically drew white letters on the screen under the photosensitive array. There were 26 populations, one for each letter, with 256 individuals each, used with a standard genetic algorithm with two point crossover and 1/300 chance of mutation for every bit. Each letter remain on the screen for about 1.5 seconds before the output filters produced an accurate reading. The effects of noise required

readings for all 26 letters several times, sharing them throughout the entire evolution over all generations. If this had not been done, each population would recognize their letter only as it was read once, without handling variance.

After a letter was displayed for the active generation, each individual in the population would load its chromosomes into the digital LLA functions and test its recognition of each letter of the alphabet. The individual's fitness was determined as a summation of the letters that were correctly identified. "Fitness" points were added for correctly identifying the letter, as well as correctly rejecting the letter, but zero points were added for incorrect identification. The points scheme prevented evolution of a premature solution by neutralizing incorrect evaluations.

4.2.1 Results

Evolving the linear functions using this setup, we were able to successfully recognize each of the 26 capital letters of the Roman alphabet, distinguishing them from all other capital letters in the alphabet.

Because the population was so large, nearly immediately there appeared some perfectly fit individuals. These individuals, though testing once perfectly, often did not test perfectly again, because there is a noise in the EAC. On an EAC that is intended to be in a steady state, the current may varies up to 0.1mA, or about 5% of the full range. To compensate for this, the current was read 12 times in a row and averaged for each letter reading. The error was further distributed over the entire population, with each member evolving in the "letter ecology" to become fit enough to recognize a specific letter.

All 26 populations ran for 70 generations, an example of the reduction in generations observed previously when configurations for exclusive-OR and the random early dropout algorithm were evolved [Deckard]. The graph for the evolution of the letter "I" is shown below (Figure 16).
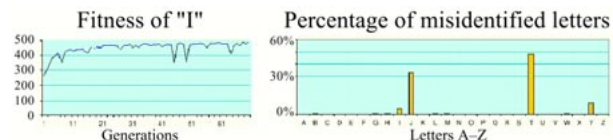

Figure 16. Evolution and accuracy of recognizer for "I"

The fitness clearly improved as each individual in the population correctly identified its own letter and 24 of the other letters repeatedly, only misidentifying two. Some of the populations did not learn to identify their letters as successfully as the "I" populations (two misidentifications). In some cases this is because the letters are similar to many other letters, like "R", "D", and "B". In other cases this may have due to the light for a critical portion of the letter not shining directly onto a photoresistor.

The average fitness of the populations at times dipped drastically down for a generation. This is because the EAC is very sensitive to changes in the environment; even turning on fluorescent lights in another room was observed to slightly change the behavior. In the earlier fitness generations the reading from the letters was affected by noise, but the genetic algorithm was able to compensate. The results of this research are a step toward more complicated image recognition

problems using an EAC, such as recognizing a DDoS "alphabet" (Section 4.3.4 Results Using Ring Method).

The results show how GAs can evolve EACs for difficult tasks. Analog computers are not encumbered in their structure by serial computations. It is natural in analog computers to compute in parallel because they compute using the parallel physics of the classical world. To solve a problem such as image recognition, an appropriate model can solve a massively parallel problem nearly instantaneously.

4.3 Detecting Distributed Denial of Service Attacks

The ability of the EAC to recognize patterns very rapidly, especially in silicon VLSI, suits it to the task of detecting distributed denial of service (DDoS) attacks. This application was first proposed to Nortel as an embedded system in a core router, which would also implement quality of service (QoS) traffic management [Mills Nortel].

Regular, desirable traffic can be characterized by a set of patterns. Traffic during a DDoS attack will generally not behave the same way as legitimate traffic. By quickly discovering an attack in progress, evasive measures could be taken to limit its effectiveness or stop it all together. One method for combating an attack would be to implement connection push-back [3] in core routers, preventing the attack from reaching the intended victim.

4.3.1 Approach

Two approaches were used to detect DDoS attacks. In the first, inputs corresponding to the traffic queues were positioned linearly at opposing ends of the board (figure 16).
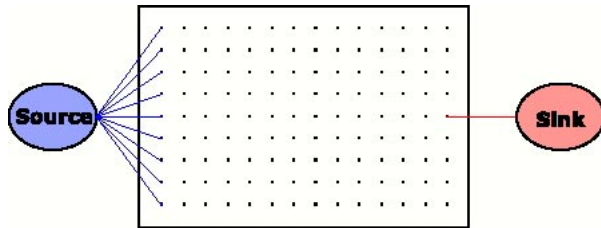

Figure 16. Linear configuration method

In the second, a ring of queue inputs were placed with an LLA in the center (Figure 17)
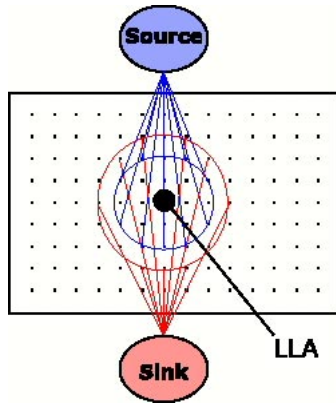

Figure 17. Ring configuration method

4.3.2 Traffic Simulation

In order to evaluate both configurations, a distribution of IP addresses determined the amount of current to apply to the EAC. The source IP addresses from a packet capture were hashed into one of eight bins. After 50 source addresses were placed into the bin, current proportional to the number of entries in each bin were written to corresponding source and sink points on the board. The packet capture we originally intended to use was for the Abilene backbone network [9]. Unfortunately, this capture was not representative of even "normal" traffic due to its high degree of private IP addresses. Therefore the normal traffic patterns were modeled by selecting random numbers between 0 and 7 inclusively and distributing these into their respective hash bins. To simulate a denial of service attack, we biased the random selection to have the fifth bin populated an additional 10% of the time. This was equivalent to seeing the same IP address more often, resulting in a higher bin count (Figure 18).
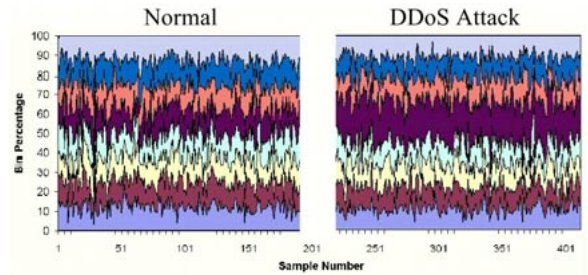

Figure 18. Normal traffic vs. DDoS attack traffic

4.3.3 Results Using Linear Method

The key difference between the two approaches was the amount of digital post-processing required to evaluate the EAC's output. While the linear method does not require the use of an LLA to perform the computation, it requires substantial digital analysis of the EAC's output. The ring method, on the other hand, does not require this digital computation because the LLA was intended to handle the determination of whether or not a DDoS attack was detected.

Using the linear method, normal Internet traffic appeared as a relatively steady gradient from a source to its corresponding sink at the opposite end of the board (Figure 20).
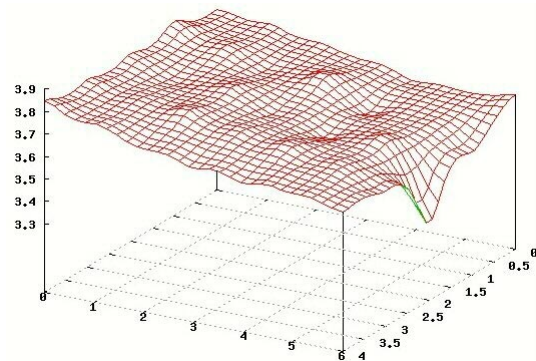

Figure 20. Output of linear method for normal traffic

During the simulated DDoS attack, a disproportionate amount of current flowed out of a given source and created a local maximum (Figure 21).
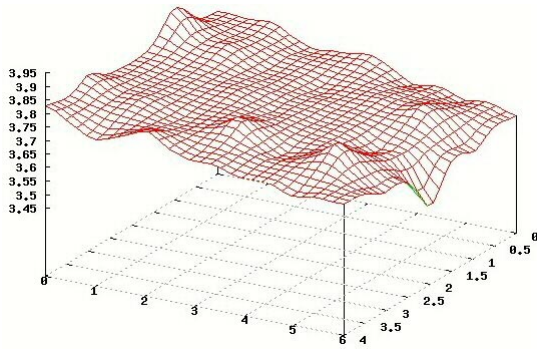
Figure 21. Output of linear method during DDoS attack

The gradients were read back from the EAC and processed by a digital computer to determine if an attack was in progress, based on the values sampled from the gradient manifold.

4.3.4 Results Using Ring Method

The ring method was less sensitive to the dimensions of the EAC board. To configure the board, we chose a point near the center of the EAC and connected that point to a Lukasiewicz logic array (LLA). We then formed a circle of queue inputs with this LLA as its focus. An outer ring of sinks which are equally spaced from their matching source ring points were added to dissipate input currents locally.

Using this configuration, normal Internet traffic would result in the sinks absorbing the majority of the source point's current (Figure 22; note scale of graph).
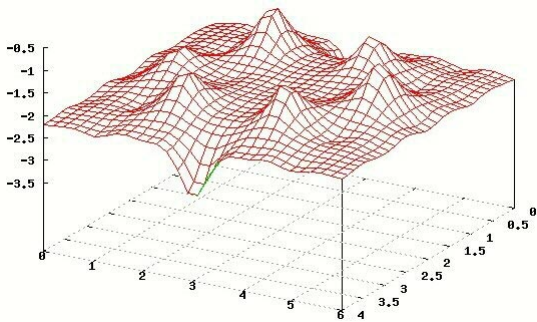

Figure 22. Output of ring method for normal traffic

When faced with a denial of service attack, the simulated DDoS traffic produced a noticeable spike was in the graph, reflecting the abnormal traffic from that queue (Figure 23; note scale of graph).
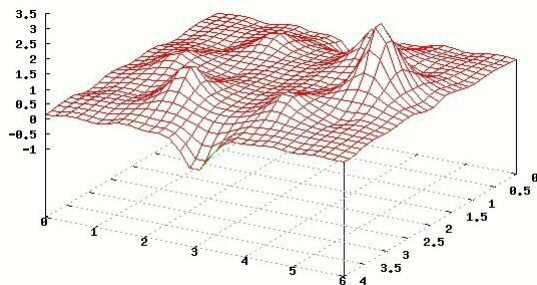

Figure 23. Output of ring method during DDoS attack

Unfortunately, the *center* of the ring was still flat, indicating that this abnormal distribution of source current did not create a substantial imbalance in the center, although it did create a recognizably different pattern. These results indicate that a single LLA positioned in the center of the ring would not be able to distinguish normal traffic from DDoS attack traffic.

However, the difference in these patterns is sufficient to be recognized by adding more LLA fuzzy logic units. DDoS attack detection can be viewed as an adaptation of the single-pixel retina [Mills] or an evolvable image recognizer (Section 4.2). In its simplest form, inputs that do not generate a balanced, low-intensity ring of traffic peaks—the letter "O"—would represent abnormal traffic. A better approach is to design an "alphabet" of traffic peaks to ascertain which queues are receiving the abnormal traffic (Figure 24).

Figure 24. Internet traffic "alphabet"

Adding additional EACs to accumulate packet addresses in multiple bins over a period of time could be used to detect flash crowds as well as DDoS attacks if the results were communicated with the other boards. Thus we recognize the importance of the original continuous retina developed by Mills [Mills] for pattern recognition, applying it to "look" at Internet traffic instead of letters in the "butterfly alphabet."

4.4 The Variety of Applications Prototyped

The two application described here in detail are only two of the many that have been studied and developed over the past six years. Students have configured extended analog computers to solve problems in a wide range of categories.

Pattern recognition: recognize commercial airliner silhouettes, recognize "Captcha" disguised text, identification of images of galaxies, and data mining as pattern recognition.

Artificial intelligence: use genetic algorithms to evolve neural network models for exclusive-OR, model gait generators for walking and flying, evolve simple artificial organisms whose actions are specified by the McCulloch-Kilmer-Blum RETIC model of behavior generation, model stereausis in the Barn Owl, develop artificial tissues to embody an artificial organism (*Tyto computatrix,* the electronic barn owl project).

Analog models of algorithms: generate random numbers (a weak form of super-Turing computation), evolve random early dropout (RED) algorithms for queue management in Internet routers, solve small instances of the NP-complete problem Hamiltonian Circuit by "looking" at graphs, model digital error-correcting codes as recurrent systems, generate tones and noise as outputs of the EAC, and study dynamical and chaotic systems.

Biological and scientific computing: model neuronal avalanching as a function of physical randomness, model weather systems, model Lindenmayer systems of plant growth, explore simple models of protein folding, render three-dimensional images using a method similar to radiosity, model galactic evolution.

Where do we intend to go from here?

5. A DISTRIBUTED ANALOG SUPERCOMPUTER PROTOTYPE

After developing these applications, it became apparent that problems suitable for solution by an extended analog computer are characterized by a need to process data in the terabyte or petabyte range, or that involve systems of thousands of partial differential equations, that yield answers which can tolerate some degree of imprecision, that are best displayed visually, and that must be computed quickly. In other words, the EAC is suited to solve Grand Challenge problems in high-performance computing.

One example of such a problem is protein folding. In nature, proteins fold in microseconds. On a digital computer, searching out possible configurations may take hours or even days. This is because the molecular components of proteins and their bond interactions are modeled using electron repulsion integrals. Solving these integrals is time-consuming. However, on an EAC, the electron clouds are represented by gradient manifolds. These model local and global interactions between parts of the protein as it collapses into a stable form.

Computation speed is even more important when real-time weather prediction models are used to identify the emergence of tornadoes in rapidly-changing weather systems. With their ability to "look" at patterns of sensor data in real time, a network of EACs could provide timely warnings of tornadoes as well as DDoS attacks. These examples motivate our design of the distributed analog supercomputer prototype (DASP).

5.1 Design of the DASP

The hardware design was simple. The factors that led to the implementation of the USB networked EAC (Section 3.3.3) produced a node that is easily fabricated and that can be used either in a local network or over the Internet. Inexpensive USB hubs attached to a digital host create useful subnets that can be expanded as funds become available (Figure 24).
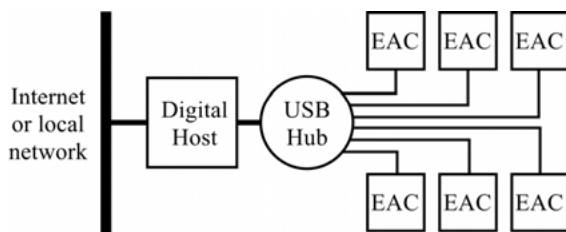

Figure 25. DASP Subnet

This concept has proved attractive. Researchers in the field of unconventional computing have supported the DASP. We have proposed a 64-node DASP cluster with a 1,600-point output array. The network shown below will cost US$16K (Figure 25). We are now seeking funding while we continue to implement the DASP network on an *ad hoc* basis.
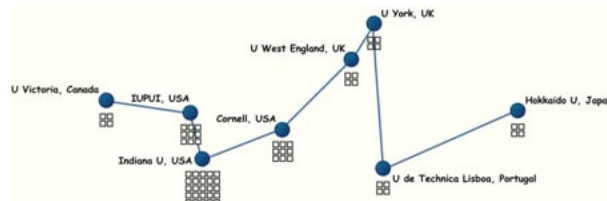

Figure 26. Proposed DASP network

5.2 Diffusion of Innovative Artifacts

Mentioning this is a matter of some delicacy, but we have found that it is more difficult to obtain a small amount of funding than it is to obtain a much larger amount of funding. However, another phenomenon is recurring that may overcome the funding difficulty. We saw it once before in 1992 when a small hexapod robot, Stiquito, was invented by the primary author. It was so small and inexpensive that it was used widely, even though it was difficult to build. Now, 14 years later, tens of thousands of these small robots have been purchased. The most recent Stiquito book contains a microprocessor controller as well as a robot kit [Conrad].

The "Stiquito phenomenon" is an example diffusion of innovation, in this case an inexpensive artifact [Rogers]. It is similar to the spread of the worldwide web, which began when free web browsers were released for use on the Internet. As we distribute nodes in the DASP, we may be watching this kind of intellectual diffusion repeat itself. Faculty at a few universities in the US, the UK and Canada have either purchased EACs or have had earlier versions donated to them. Even a group of students at the University of Illinois at Urbana-Champaign has purchased several EACs for their biocomputing research. If several hundred DASP nodes or subnets become connected to the Internet, portal software that runs in the background of workstations, similar to that used in the SETI screensaver, could permit users to share DASP computing time. The next generation supercomputer might not be funded by any single agency or institution at all, but by users with a few hundred or a few thousand dollars available!

In the long run, the design of wafer-scale (or at least multi-square-centimeter VLSI chips) fault-tolerant arrays of silicon VLSI EACs will allow us to implement supercomputers such as the DASP in a workstation or laptop. Such devices might solve trillions of partial differential equations per second This research goal, as inexpensive as it is, will open the door on a completely new generation of computer architectures.

5.3 Towards One Trillion Partial Differential Equations Per Second

In 1994 the response time of an early version of a silicon retina was measured at two nanoseconds as a laser beam was swept across it [Biswas]. Conductive plastic foam is slower, with a response less than a microsecond based on input from a frequency generator. Maximum throughput for an EAC solving a Laplacian partial differential equation thus range from 10^6 to 10^8 PDEs per second after configuration.
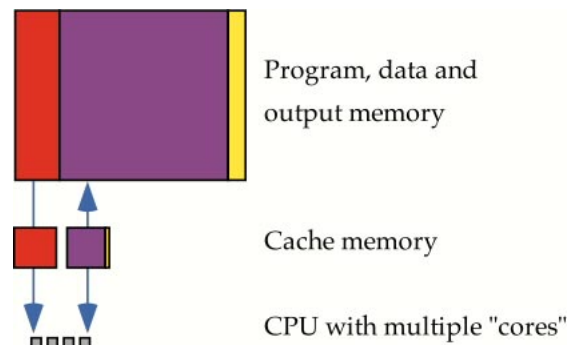

Figure 27. One-dimensional von Neumann bottleneck

To get a sense of the potential performance of the DASP, the fundamental architectural difference between the EAC and a digital computer is presented. In a digital processor, computation is limited by the one-dimensional von Neumann bottleneck, which limits parallelism as instructions are fetched from memory (Figure 27). But in the EAC the "Rubel" bottleneck is two-dimensional. Once the machine is configured, which can be done in parallel if one digital processor is assigned to each input-output point, the EAC can process a sequence of two-dimensional inputs. Data is streamed through the device in a manner reminiscent of a digital signal processor (Figure 28).
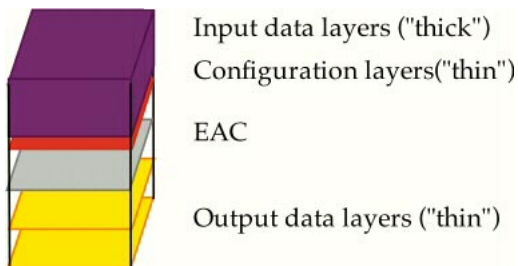

Figure 28. Two-dimensional "Rubel" bottleneck

If implemented in silicon, the DASP, with 64 two-dimensional "bottlenecks" would solve approximately 3×10^{11} PDES per second. Performance would be limited by data availability and even packet flight time through the Internet. Yet thousands of nodes and subnets could increase grid performance into the range of trillions of PDEs per second.

6. CONCLUSION

At the keynote address for the 2002 International Symposium on Computer Architecture, Bob Colwell of Intel predicted that the failure of Moore's Law would lead to a new epoch in computer architectures and systems. He attempted to predict what the new epoch might bring, but admitted that any predictions would be unlikely to match the future. He was right. While it is impossible to accurately predict the future—magazines from the 1950's were full of flying automobiles, for example—ten years ago as we began this research it was not anticipated that Rubel's extended analog computer would lead to operational Internet-accessible prototypes costing only $250 apiece, about $1/6$th of the cost of a modern workstation PC. It was out of our conception to predict that the EAC would result in a distributed supercomputer. The scope of this modern paradigm for analog computing has grown as we gained experience with it.

Even though we still do not know what the future holds, we are working to create it. We believe that the need for robust, efficient, inexpensive, flexible and fast computer architectures will not go away, and that Rubel's extended analog computer is a straightforward way to satisfy it.

7. ACKNOWLEDGEMENTS

8. REFERENCES

[1] Blair, B., Debray, S. K., and Peterson, L. L. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst., 15,* 5 (Nov. 1993), 795-825.

[2] Conrad, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[3] Deckard, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[4] Girvin, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[5] Graca, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[6] Himebaugh, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[7] Karplus, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[8] Maass, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[9] McNaughton, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[10] Miller, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[11] Mills, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[12] Mills, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[13] Nijhout, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[14] Pour-El, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[15] Rubel, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[16] Rubel, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[17] Sandved, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.

[18] Shannon, W., and Marchionini, G. *A Study on Video Browsing Strategies.* Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD, 1997.