Polymer Processors

Jonathan W. Mills
Computer Science Department
Indiana University
Bloomington, Indiana 47405

Abstract

A recent paradigm for analog computation is suited to conductive organic polymers. Partial differential equation solvers and continuous-valued logic cells implemented as conductive sheets and diode arrays have canonical structures that can be reconfigured digitally. An example application, a pattern-recognizing sensor, is presented as a general example of a polymer processor.

1. Introduction

The architecture of polymer processors is based on a recent theoretical model of analog computation called the extended analog computer [Rubel 1993]. This model is not a general purpose analog computer (GPAC) [Shannon 1941], nor is it one of the many analog neural network models implemented in silicon VLSI [Mead 1989] and studied theoretically [Siegelman/Sontag 1992]. It is a new paradigm for computer architecture that even recently was believed to have no physical realization [Campagnolo/Moore/Costa 2000].

But a general, and probably universal, extended analog computer exists with an electronic implementation. It requires a new design methodology, and new ways to think about compiling "programs", in this case configurations overlaid onto a standard analog machine. The extended analog computer can be implemented with both silicon VLSI and conductive polymers. Conductive organic polymers were discovered by condensed matter physicists Heeger, MacDiarmid, and Shirakawa. The polymer field-effect transistor was developed by Garnier. Polymer processors use these technologies in simple but novel ways to solve a broad range of problems more rapidly and efficiently than digital computers. While polymer processors cannot efficiently implement symbolic applications, such as word processing, for those applications within their scope they are high-performance, inexpensive non-silicon computers.

2. Theoretical Model

The *general purpose analog computer (GPAC)* is the mathematical model of what is usually thought of when an "analog computer" is mentioned. The archetypal analog computer in this class is the differential analyzer, a mechanical computer invented by Vannevar Bush of MIT. Using various gears, curve plotters, and cams, the differential analyzer solved systems of ordinary differential equations. Several years later, advances in circuitry permitted electronic differential analyzers to be built. These analyzers used integrators (operational amplifiers), constants, multipliers, and addition circuits in application-specific configurations to solve ordinary differential equations. However, GPACs could not directly solve systems of partial differential equations. These problems, known as field problems, were solved with various conductive solids, liquids, or resistive meshes [Karplus 1958]. After the advent of digital computers, hybrid digital-analog computers solved field problems by iterating "across" the field, varying the boundary conditions with each trial.

The *extended analog computer (EAC)* is composed of a general purpose analog computer to which is added a "quintessential boundary-value problem solver". In practice this is a partial differential equation solver. Those partial differential equations that are strictly boundary-value problems are Laplacian; those that have internal conditions are Poisson. Many problems described by these equations, such as heat diffusion and internal mechanical stress, may be solved electronically [Karplus 1958]. Observations that vertebrate brains solve field problems to generate spatio-temporal maps led

Rubel to extend the GPAC with functional elements that performed field computation directly [Rubel 1985, Rubel 1993]. Rubel's assertion that *"...the problem of implementation is to describe physical, chemical, and/or biological devices that do in the real world what the black boxes are supposed to do. Since the EAC is so broad, this can never be done in practice...",* suggested that no practical extended analog computer architecture would be possible. However, research into the continuous-valued Lukasiewicz logic as a computational paradigm led to an electronic implementation of the extended analog computer.

Lukasiewicz logic has a denumerably infinite number of truth values. It describes the class of ideal analog circuits that have an infinite maximum precision. Real analog circuits, which have a finite maximum precision, are classified by the number of data values encoded on individual wires in the circuit. Boolean logic is the most familiar subset of Lukasiewicz logic: it describes binary digital circuits. Discrete multiple-valued circuits are described by logics with $2 < n \leq 16$. Continuous-valued analog circuits are described by subsets of Lukasiewicz logic with $2 \leq n \leq 2^p$, where p is the maximum precision of the circuit in bits. Typically p falls in the range $6 \leq p \leq 12$.

A *Lukasiewicz logic array (LLA)* is an H-tree array that implements a schema for sentences of Lukasiewicz logic [Mills/Beavers/Daffinger 1990]. The processing elements of the array correspond to either implications or negated implications in the sentence schema (Figure 1). Lukasiewicz implication is defined by the valuation function min(1, 1-a+b). Negated implication has a valuation function defined as max(0, a-b).

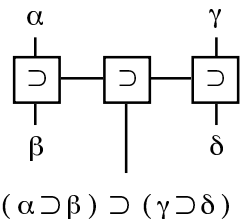

Figure 1. Sentence in Lukasiewicz logic and corresponding H-tree array

Lukasiewicz logic arrays approximate GPACs sufficiently well for practical purposes. An obscure theorem validates this substitution by proving that algebraic ordinary differential equations—the class of equations solved by GPACs—are approximated arbitrarily closely by sentences in Lukasiewicz logic [McNaughton 1950].

Although Rubel was unable to restrict the extended analog computer to a Turing-complete model, and believed that it might be too powerful to be mathematically interesting, other theorists have shown that some variants of the GPAC on which the extended analog computer is based, when extended in only one dimension, are no more powerful than Turing machines [Campagnolo/Moore/Costa 2000]. In general, in the presence of noise, no analog computer can be more than Turing-complete [Maass/Sontag (no date)]. From the view of a computer architect these results are more reassuring than anything else, given the many claims in the early 1990's that analog computers were computationally more powerful than Turing machines. Implementing useful extended analog computers is enough of a challenge.

3. Architecture and Implementation

The high-level architecture of a polymer processor is a restricted form of extended analog computer. It is composed of some number of levels with a "lot" of feedback. The theoretical model has a finite but unbounded number of levels. In practice two or three layers of partial differential equation solvers operating in parallel are sufficient to implement neural networks, feedback controllers, and active sensors.

Each level is composed of a conductive sheet and a Lukasiewicz logic array (Figure 2). Relatively few interconnections are necessary, a benefit for both polymer and silicon VLSI designs.

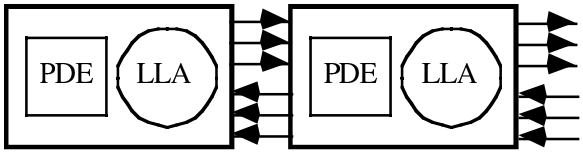

Figure 2. Multiple-level extended analog computer architecture

The original Lukasiewicz implication cell used twelve transistors (Figure 3a). By applying algebraic transformations to this circuit, it was reduced to a single diode (Figure 3b) [Mills/Beavers/Daffinger 1990, Mills 1998]. It has been used as a novel edge detector that not only finds horizontal, vertical and diagonal edges, but also encodes their orientation [Mills 1992, Mills/Walker/Himebaugh 2003].

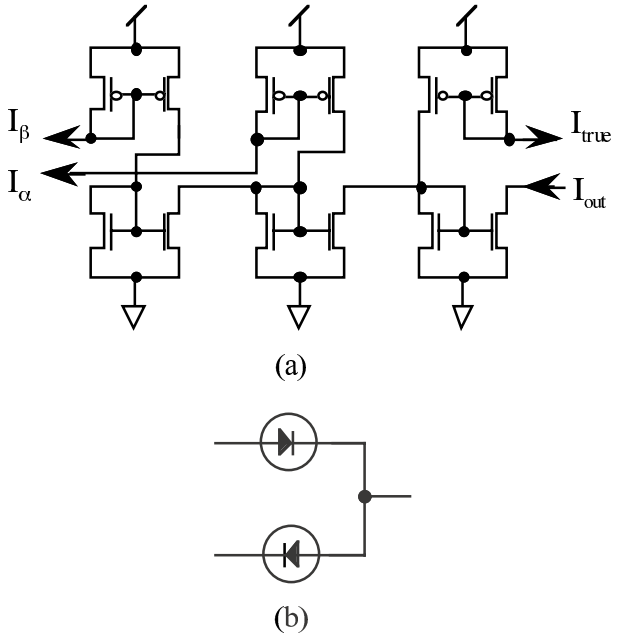

Figure 3. (a) 12-transistor implication and (b) pair of diodes for two implications

In a digital computer, wire is necessary to move data and instructions between memory and the central processor. This creates the well-known von Neumann bottleneck. In a polymer processor computation is pervasive, and often even the wires compute. While some parts of the machine may not contribute to the computation, it is unusual for any part to restrict it. For example, the pattern-recognizing sensor merges detection and recognition, and computes continuously over the entire image (Section 5, Applications). This device has been built in silicon and simulated in conductive foam. Transparent conductive polymers are desirable for this kind of processor because the sensor and recognizer can be stacked vertically, unlike the silicon VLSI version, and the output displayed through the transparent polymer on the input image.

The components of a polymer processor fall into two categories. The first category is computational: partial differential equation solvers, piecewise linear function generators, and wire junctions that perform addition and subtraction. The partial differential equation solvers are implemented as conductive sheets. Piecewise linear function generators are implemented as reconfigurable Lukasiewicz logic arrays, which are often nothing more than binary trees of diode-connected field-effect transistors. Addition and subtraction is performed with wire junctions, and is governed by Kirchhoff's current laws. Components in the second category manipulate data. Current mirrors duplicate values for use in multiple locations. Wires explicitly pass parameters between spatial locations in the machine. Conductive sheets serve to communicate values and simultaneously scale their sum. A conductive foam prototype of a polymer processor that uses discrete silicon components for the Lukasiewicz logic array is publicly available on the world-wide web at:

http://cgi.cs.indiana.edu/~bhimebau/eac.cgi

It solves a Poisson partial differential equation. The solution is used to emulate a network of leaky neurons that compute exclusive-or.

4. Programming and Compiling

All computers have semantics that describe their structure and operation. In a digital computer these semantics may be treated *independently*. They are related by clearly defined interfaces between semantic boundaries. Ignoring much detail, there are two semantics, one for the architecture and another for the program. These semantics are usually independent for most applications. This is a strength of a digital computer. In principle, a program can be written that will run on many different architectures. In practice, this is not as simple as it sounds.

In a GPAC the two semantics are *indistinguishable*. In this kind of analog computer, the hardware is the software, and vice versa. To change the program, one changes the machine. In the 1960s, analog computer "programs" were separated from the hardware. The user carried a part of the machine, called a

"patch board" because it was a tangle of wires and components interconnected on a circuit board, from the office to the computer where the "program" was run. The wiring on the patch board was changed if an error was detected. After each use the patch board was disconnected and returned to storage. GPAC programs were cumbersome, at best.

The structural and functional semantics of extended analog computers are *interdependent*. The canonical structure of the machine provides a general framework for many different computations, but the hardware is standardized like a digital computer rather than a GPAC. The function of the extended analog computer is carried out simultaneously at many points in the architecture, like a GPAC, but these points are easily reconfigured, in the manner of a digital computer. Furthermore, the reconfiguration can be controlled by a vector of bits, similar to a program for a digital computer. The configuration is called an overlay, expressing the idea that the function of the machine changes over all components, rather than in a restricted location—the memory—of a stored program digital computer.

An overlay for a polymer processor is developed differently than a digital computer program. The entire overlay must be created, not parts of it that leave the rest unimplemented. This is similar to the manner in which reconfigurable digital processors are designed, for example, application specific field-programmable gate arrays. Digital computers automatically generate overlays for polymer processors using genetic algorithms and a fluid model of the conductive sheet (similar to the fluid model of the field-effect transistor). In a research seminar conducted at Indiana University in the fall of 2002, a canonical extended analog computer was described using a genome for the overlay. Genetic algorithms were developed by student researchers for several diverse applications. The results were encouraging. Starting with populations of 10,000 individuals, each a randomly generated overlay, the chosen applications were evolved within 1000 generations. But recognizing the standard structure of the canonical processor, and observing that an analog computer has a strong correlation between the problem semantics and the overlay, the students reduced the population to 1000 individuals, the number of generations to three, and the time to find a solution to several minutes on a Pentium-class laptop. These results are still being explored. One possibility is to dynamically evolve control overlays for polymer processors used as embedded controllers.

5. Applications

The first applications for extended analog computers were difficult to design. For example, finding an implementation of exclusive-or using an extended analog computer took over one year. This application is significant because exclusive-or is the simplest non-linearly separable function that distinguishes general neural networks from perceptrons. In retrospect the solution was simple, but there was no existing frame of reference within which the problem could be understood [Hedger 1998]. By the fall of 2002, a few problems had been solved, some using conductive plastic.

All applications had a canonical form derived from the theoretical model of the extended analog computer. One application, a "single-pixel" pattern-recognizing sensor, appears in various guises in a cyclotron beam line controller, an internet router using the Random Early Dropout algorithm, and real-time ceramic spall detection in lubricating systems. This application will be discussed in detail.

The pattern-recognizing sensor is a one-level extended analog computer (Figures 4 and 5).

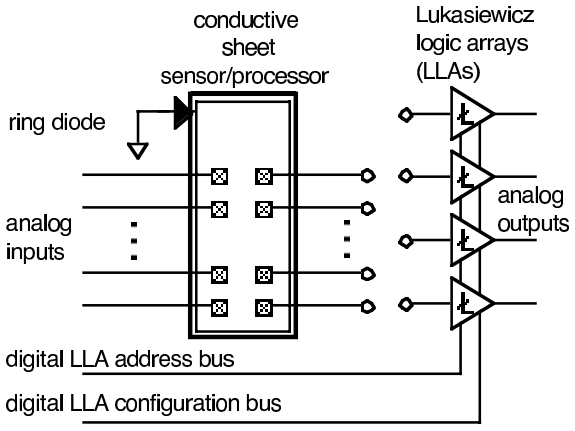

Figure 4. Pattern-Recognizing Sensor Schematic

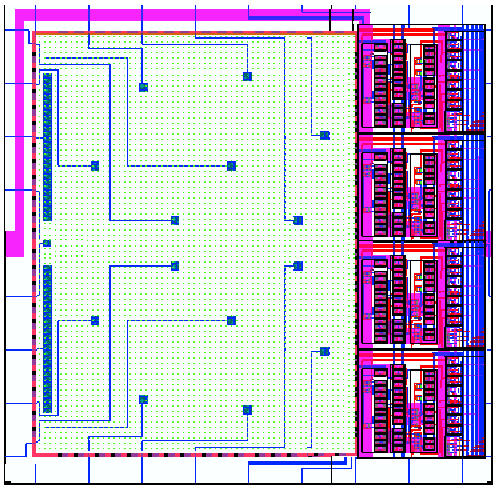

Figure 5. Pattern-Recognizing Sensor

When implemented in silicon VLSI, the conductive sheet and the Lukasiewicz logic arrays are adjacent to each other. Because silicon is not translucent, the original image must be projected onto the sheet, and cannot pass through it. Nor can multiple layers be stacked, with communication flowing between levels, either electrically or optically. Silicon VLSI is planar, although nanotechnology is being investigated for three-dimensional processors) [KleinOsowski/Kiehl/Lilja 2002].

The pattern-recognizing sensor operates as a trainable character recognizer, a task often performed by neural networks. A simple recognizer that distinguishes between "A" and "T" illustrates how the sensor is trained using the piecewise linear functions generated by a Lukasiewicz logic array. A binary encoding for the characters is chosen, with "A" indicated by 111 and "T" by 000. The functions are initially set to produce a 1 for every input to the sensor. Initially presenting an "A" does not generate an error, so no adjustment is necessary (Figure 6).

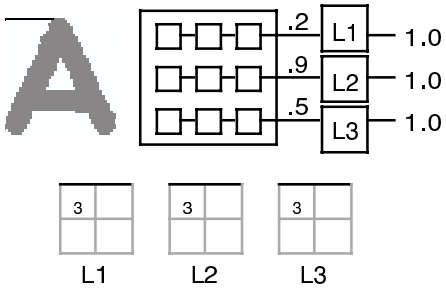

Figure 6. Initial presentation of "A"

When the "T" is presented, an error results, which can be used with the desired output and the values input to the Lukasiewicz logic arrays to determine the new functions. A simple rule for choosing the functions is to minimize the error between the current presentation and the most recent but different character presented. Following this rule, the new functions are shown in Figure 7.

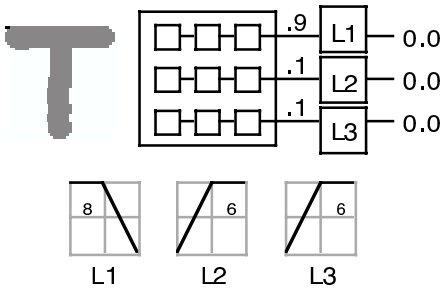

Figure 7. Presentation of "T"

Presenting "A" again indicates that this choice of function settings is acceptable. Finally, the ability of the sensor to classify characters as either "A", "T", or similar to one or the other is tested with four images: a scrawled "A" and a "T" with a circular patch removed (Figure 8).

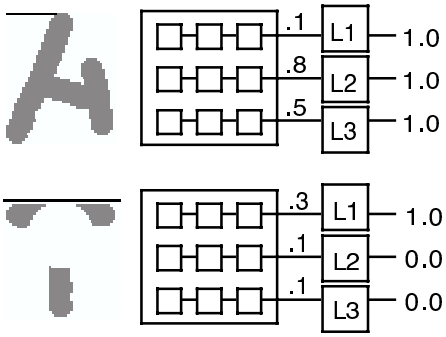

Figure 8. Categorization of similar characters

The recognizer is robust because of the continuity in the induced current gradient and the piecewise-linear function units. The conductive sheet is inherently capable of smoothing discontinuities in the image.

A polymer recognizer improves on this design because it can be made translucent. Such a computational sensor could "read" an image,

detect oriented edges and masses within it, and recognize patterns, all using a series of thin layers. Anomalies on the original image could be detected and displayed on top of the visible image using LEDs built into the polymer sheet. Such a recognizer, large enough to cover an entire image, such as an X-ray, could be held up to the image while it performed these functions in real time. At best such a device would be a difficult for silicon digital computers to duplicate, and in some cases would be impossible to implement with the form factor described (thin transparent sheets). A device such as this is only one possibility within the scope of today's conductive polymer technology and the computational paradigm offered by extended analog computers.

6. Summary

As our understanding of polymer processors continues to improve, applications that are computationally challenging for digital computers are emerging. Most recently, radiosity-based image rendering has been simulated [Olowoyeye 2003]. The future of polymer processors looks bright.

7. References

[Campagnolo/Moore/Costa 2000] Iteration, Inequalities, and Differentiability in Analog Computers. Santa Fe Institute. http://www.santafe.edu/~moore.

[Hedger 1998] Analog Computation: Everything Old is New Again. Indiana University Research & Creative Activities, [XXI:2], pp. 24-27. http://www.indiana.edu/~rcapub/v21n2/p24.html

[Karplus 1958] Analog Simulation, McGraw-Hill: New York.

[KleinOsowski/Kiehl/Lilja 2002] Fault-Tolerant NanoBoxes for Designing Computers Using Nanotechnology. First Workshop on Non-Silicon Computation.

[Maass/Sontag (no date)] Analog Neural Nets with Gaussian or other Common Noise Distributions Cannot Recognize Arbitrary Regular Languages. Email: maass@igi.tu-graz.ac.at.

[McNaughton 1950] A Theorem About Infinite-Valued Sentential Logic. Journal of Symbolic Logic, 16 (1950), pp. 1-13.

[Mead 1989] Analog VLSI and Neural Systems, Addison-Wesley: Reading, Massachusetts.

[Mills/Beavers/Daffinger 1990] Lukasiewicz Logic Arrays. Proceedings of 2oth International Symposium on Multiple-Valued Logic. Charlotte, North Carolina: IEEE Press.

[Mills 1992] Area-Efficient Implication Circuits for Very Dense Lukasiewicz Logic Arrays. Proceedings of 22nd International Symposium on Multiple-Valued Logic. Sendai, Japan: IEEE Press.

[Mills 1998] U.S. Patent 5,770,966. Area-Efficient Implication Circuits for Very Dense Lukasiewicz Logic Arrays.

[Mills/Walker/Himebaugh 2003] Lukasiewicz' Insect: Continuous-Valued Robotic Control after Ten Years. Int. Jour. Multiple-Valued Logic.

[Olowoyeye 2003] Image Rendering using Wave Theory of Light and Extended Analog Computers. Indiana University Computer Science Department Technical Report 581.

[Rubel 1985] The Brain as an Analog Computer. *J. Theoretical Neurobiology* **4** (July 23).

[Rubel 1993] The Extended Analog Computer. Adv. In Appl. Math. 14, 39-50.

[Seigelmann/Sontag 1992] On the Computational Power of Neural Nets, Proc. Fifth ACM Workshop on Computational Learning Theory, Pittsburgh, July 1992, pp. 440-449.

[Shannon 1941] Mathematical Theory of the Differential Analyzer, J. Math. Phys. MIT, 20 (1941) pp. 337-354.