An Analog Accelerator for Linear Algebra[1]

Yipeng Huang, Ning Guo, Mingoo Seok, Yannis Tsividis, Simha Sethumadhavan

In our ISCA 2016 paper, we discussed results of a five-year, full-system design and prototyping effort aimed evaluating the benefits of the analog computing paradigm in a modern computing context. Analog computing breaks two fundamental and pervasive assumptions in computing today: first, instead of using digital binary numbers, the analog computational model encodes numbers as continuous values. This means that a datapath for an FP operation can be reduced to a single wire. Second, in the analog computational model, there is no need to break a problem into discrete steps, *i.e.*, iterative digital algorithms, such as the Newton-Raphson algorithm for solving ODEs, are not needed in the analog model. Avoiding algorithmic discretization avoids many intermediate steps, mainly memory accesses, thereby lowering memory costs. In theory, these advantages can offer several orders of magnitude improvement over digital computing.

The first step in the project was to select an application domain to study. We thoroughly reviewed prior work on analog computing from the 1950s and 60s. In that era, analog computers were advertised as standalone general-purpose computers. Today, we would probably describe them as standalone calculators. These calculators were mostly designed to solve projectile trajectories, or more generally, ordinary differential equations. In contrast, in the modern computing context, most problems in scientific computing and machine learning are not solved using ordinary differential equations; instead, problems are cast and solved as linear algebra problems. So we decided to study how to solve systems of linear equations in the analog computational model.

As a review, systems of linear equations are solved in digital computers using either direct or iterative numerical methods. Direct methods such as Gaussian elimination factor the systems of linear equations, obtaining the components of the solution one-by-one. On the other hand, iterative methods start with an initial guess for the entire solution vector, and update the solution vector over several iterations of the algorithm, each step minimizing the difference between the guess and the correct solution. Iterative methods are increasingly important due to the appealing fact that intermediate guess vectors are a good approximation of the correct solution. It appeared to us that analog computing would be effective for this domain if we can avoid the iterative solutions by finding a good analog algorithm and analog hardware to execute the problem. Since we did not have either we decided to co-design both.

Architectural requirements: Our co-design requirements were simple: we favored flexibility/programmability wherever possible, and were extremely risk-averse even if it meant making some suboptimal design choices. The thinking was that if the computational model can provide orders of magnitude improvement, a few percentage points wouldn't matter. Further, we decided that we had to prototype the chip, because unlike digital systems where place and route (or even simulations in some cases) gives high confidence, analog systems are prone to noise and require prototyping (at least to get some initial idea

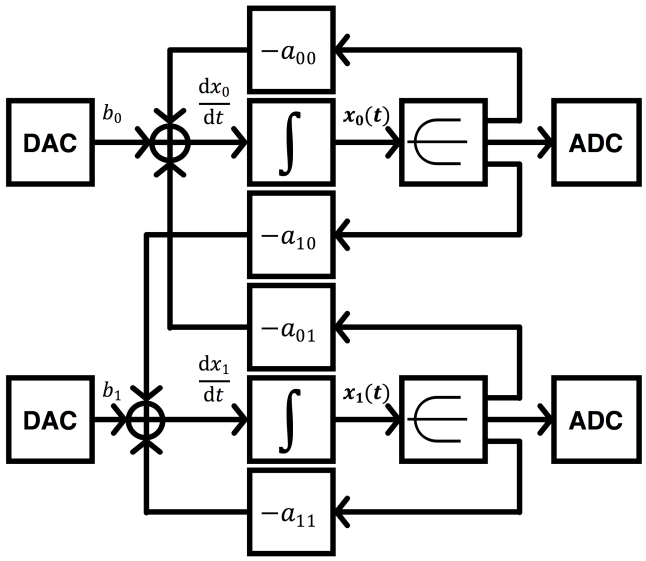

Figure 1. Schematic of an analog accelerator for solving Ax=b, a linear system of two equations with two unknown variables.

of noise bounds). These two high-level requirements drove a lot of our design decisions, and, down the line, also limited what we can and cannot study on the prototyped chip. Finally, we also had a limited budget (2 PhD students on an NSF medium budget); so we had limited resources in terms of the size of the chip.

The flexibility/programmability requirement meant a few things in the context of this analog computing project: 1) an ability to report to the programmer via analog exceptions when values during computations go out of range during execution 2) an ability to configure not only linear algebra problems but also a wide variety of emerging problems including those that are exceedingly difficult to solve on a digital computer and 3) abilities to individually measure and tune the individual functional units to study the effect of manufacturing and temperature variability. The last requirement was motivated by the fact the old generation of analog computers were plagued with limited accuracy.

We chose the following basic building blocks for our architecture: integrators, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), multipliers (gain amplifiers), adders, and digital memories for constants. The choice of digital memories was risky: integrating analog and digital in such close proximity had not been tried before in the computational context but the risk would be worth it as it would allow us to do transcendental differential equations (this was not used in the paper but we have follow-on work using these memories). As for the number of functional units, we were extremely constrained in the terms of the area budget, and given we had to do two iterations, we decided on the smallest number of functional units that would allow us to extrapolate the results in a meaningful manner.

Once we picked the type and number of analog functional units, the next thing to decide was the sequencing model for the analog computer. We decided to use a dataflow execution

[1] Y. Huang, N. Guo, M. Seok, Y. Tsividis and S. Sethumadhavan, "Evaluation of an Analog Accelerator for Linear Algebra," 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, 2016, pp. 570-582.

model for the analog accelerator, i.e., any problem to be solved on the analog accelerator would be expressed as a full, unrolled dataflow graph and we would simply wire up the functional units so all the computation would be done from start to finish. We created a circuit-switched crossbar connecting the functional units to others. These design decisions avoided the hassle of designing analog "memories". The accelerator off-load model seemed like a natural fit for the problem domain.

Error control was another first-order architectural design requirement. In analog circuits, subtle variations in hardware due to process and temperature variation would become variations in the computation result. We identified three main sources of inaccuracy in analog hardware: gain error, offset error, and nonlinearity. Gain and offset errors refer to inaccurate results in multiplication and summation; we added support to calibrate these individual components using additional DACs. Nonlinearity errors occur when changes in inputs result in disproportionate changes in outputs, and occur when analog values exceed the range where the behavior of the circuit is linear. We added analog exceptions to warn programmers that values have exceeded the valid range.

Mapping problem onto the hardware: The next step after the architectural design was to map the problem into hardware. In an analog accelerator, systems of linear equations are also solved similarly to the digital iterative algorithm, with an important distinction that the guess vector is updated using infinitesimally small steps, over effectively infinitely many iterations. This continuous trajectory from the original guess vector to the correct solution is actually an ordinary differential equation, which we can naturally solve using an analog accelerator!

As shown in Figure 1, at the heart our analog accelerator are integrators, which store the present guess of the solution vector. We perform operations on this solution vector by feeding the vector through multiplier and summation units. DACs provide constant coefficients and biases. Using these function units, we create a linear function of the solution vector, which is fed back to the inputs of the integrators. In this fully formed graph, the time derivative of the solution vector is a linear function of the solution vector itself. The variables stored in the integrators converge on the correct solution vector that satisfies the system of linear equations. When the analog variables are steady, we sample them using ADCs.

Our actual chip had very few integrators to directly solve any large scale problems. The way to overcome this is to use break the problem into smaller problems and map them onto the analog accelerator. Using this decomposition, we were able to estimate the benefits of analog acceleration to larger problems.

A final aspect of problem mapping was to get higher precision results (at least comparable to the digital system). We get higher precision results from the analog accelerator by running the analog accelerator multiple times. Even though the analog variables are themselves highly precise, sampling the variables using ADCs can only result in 8 to 12 bits of precision. We use the digital host computer to find the residual error in the solution, and set up the analog accelerator to solve a new problem, focusing on the residual. Each problem has smaller-magnitude variables than previous runs, allowing us to scale up the variables to fit the dynamic range of the analog hardware. We iterate between analog and digital hardware a few times to get a more precise result than using the analog hardware alone.

Discussion of results: Solving linear algebra problems using ODEs on an analog accelerator has several potential advantages, compared to using a discrete-time algorithm on a conventional computer: First, the analog accelerator uses an explicit data flow graph where the sequence of operations on data is realized by connecting functional units end-to-end. During computation, there are no overheads in fetching and decoding instructions, and there are no accesses to digital memory. The former is a benefit of digital accelerators too but the latter is a unique benefit of the analog computational model.

Second, the analog accelerator hardware and algorithm both operate in continuous time. The values stored in the integrators are continuously being updated, and the update rate is not limited by a finite clock frequency, otherwise the limiting factor in discrete-time hardware. Furthermore, a continuous-time ODE has no concern about the correct step size to take to update the solution vector, in contrast to discrete-time iterative algorithms where computing the correct step size represents the majority of operations needed per iteration. In these regards the analog accelerator is potentially faster than discrete-time architectures.

Third, the analog accelerator solves the system of linear equations using real numbers encoded in voltage and current. Each wire is able to represent the full range of values in the accelerator. Furthermore, multiplication, addition, and integration are all comparatively straightforward on analog variables compared to digital ones. In these regards analog encoding is potentially more efficient than digital, binary encodings.

We tested analog acceleration for linear algebra using our prototype analog accelerator implemented in 65nm CMOS technology, shown in Figure 2. The accelerator consists of 4 integrators, plus accompanying DACs, multipliers, and ADCs connected with a crossbar. The physical prototype validated the analog circuits and allowed physical measurement of component area and energy. Using the circuit schematics that were validated in the physical chip, we built simulation models of analog accelerator designs for extrapolation to larger designs.

We compared the analog accelerator and digital approaches in terms of performance, hardware area, and energy consumption, while varying the number of problem variables and the choice of analog accelerator component bandwidth, a measure of how quickly the analog circuit responds to changes. The digital and analog architectures solved to equal solution precision, equivalent to the precision obtained from one run of the analog accelerator equipped with high-resolution ADCs. On the digital side, we stopped the numerical iterations short of the machine precision provided by floating point numbers. We use a sparse system of linear equations derived from an elliptic partial differential equation as an example problem.

We found that an optimal analog accelerator design that balances performance and number of integrators should have components with an analog bandwidth of approximately 320 KHz. We estimated that the area and power cost increase linearly in proportion to the choice of analog bandwidth. With this conservative model, high bandwidth analog computers come with high area cost, quickly reaching the area of the largest GPU dies. Limiting the area size of the analog accelerator to 600 mm^2 limits the analog accelerator to approximately 400 integrators. On performance and energy metrics, we find that with 400 integrators operating at 320 KHz analog bandwidth, analog acceleration can potentially have 10× faster solution time and 33% lower energy consumption compared to a digital general-purpose processor.

Long-Term Impact

Pedagogy: Our paper is a concrete case study in cross-layer optimizations. We systematically discuss the difficulties encountered in analog computing, including limitations in programmability, accuracy, precision, and scalability. We present potential solutions to each of those problems across all levels in the stack from algorithm down to circuits.

The paper may even be interesting to undergraduates: Our paper explains what needs to change in computers when fundamental assumptions change. We point out that analog computing abandons the two most basic abstractions in computers: discrete-time operation and digital binary representation of numbers. Then, we elucidate how to work with these big changes.

Furthermore, in the course of the paper, we summarize an immense amount of work from an earlier history of electronic computing, distilling it to an accelerator design which may make analog computing relevant in the digital era. As the search for breakthrough computer architecture technologies is becoming more urgent, this is a good case study that any part of the computer architecture hierarchy can be reimagined, including the idea of digital computing itself.

Understanding limitations of analog: Analog computing has attracted attention in recent years, especially in the direction of analog support for neural inspired architectures. Our paper is an important contribution to the area discussing threats to analog computing, their mitigations, and limitations. We recognize the performance increases and energy savings are not as drastic as one expects when using a domain-specific accelerator built upon a fundamentally different computing model than digital, synchronous computing. We have identified two reasons for this shortfall: the high area cost of high bandwidth analog components and the challenge of beating digital at this class of workloads. Future compact high-bandwidth analog components may make analog acceleration for linear algebra a reality. In the meantime, analog acceleration holds promise in problem classes such as nonlinear systems, where digital algorithms and hardware architectures have been less successful.

Research impact: The research in this paper has captured the attention of DARPA's Analog and Continuous-variable Co-processors for Efficient Scientific Simulation program. The goal of the program is to discover computer architectures that would drastically speed up simulations in plasma physics (in the context of nuclear fusion), a type of problem ill-suited for conventional architectures. The program tests the hypothesis that changing the basic abstractions in computer architecture may change what types of problems are solvable. Interesting physical phenomena are usually continuous-time, analog, nonlinear, and often stochastic, so the computer architectures and mathematical abstractions for simulating these processes should also be continuous-time and analog. In these regards this paper may be the first in a line of work redefining what problems are tractable and should be pursued for analog computing.

Early technology transfer: A student on the project has started a company funded by DARPA to explore how analog computing model can support problems in scientific computing. That ongoing project studies the potential benefit and viability of analog accelerators for solving nonlinear partial differential equations and optimization problems.

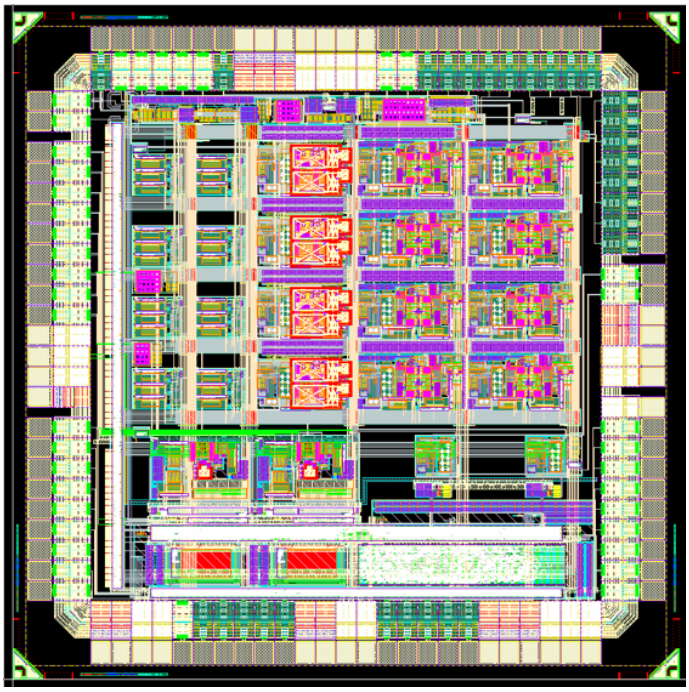

Figure 2. Circuit schematic of a physically-prototyped reconfigurable analog accelerator.

Citation of paper if it won the test of time award in 10 years?

This paper is a pioneering contribution that is discussed at undergraduate and graduate classes illustrating the benefits, limitations and engineering decisions involved in reconsidering fundamental abstractions in computing.

In addition to proposing an architecture aimed at solving long-standing problems with analog computing, the paper also focused computer architecture research efforts towards problems that are uniquely solvable using analog computing.