Hybrid Analog-Digital Solution of Nonlinear Partial Differential Equations

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

Our 2017 MICRO paper[1] explores architectural ideas that allow us to successfully use a prototyped analog accelerator, here in the context of scientific computing. We use the approximate solution from an analog accelerator to help a precise digital nonlinear equation solver running on a GPU. For larger, more nonlinear problems, the hybrid analog-digital solver has a performance improvement of 5.7× and energy savings of 11.6×, relative to a GPU without the help of an analog accelerator.

The favorable findings contrast with our prior work[2] which found analog accelerators would have limited benefits in solving *linear* problems, due to prohibitively high analog silicon area costs and due to stiff competition from efficient digital computer algorithms for linear algebra. Here in this year's paper, analog acceleration redeemed itself in *nonlinear* problems, which pose no special challenge in analog but are tricky in digital because the prototypical digital algorithms for nonlinear equations are unreliable.

The difference in how digital linear solvers vs. digital nonlinear solvers spend computation time is a key factor why analog acceleration had limited impact in helping with linear equations, while significantly helping with nonlinear problems. Digital *linear* solvers give the most significant digits of the solution quickly but take time to give the least significant digits, and unfortunately analog acceleration cannot help to give precise solutions. On the other hand, digital *nonlinear* solvers have a hard time getting a rough-guess solution but polishing a good guess to high precision is cheap. A hybrid analog-digital system combines the analog solver for cheap approximate solutions and the digital solver to obtain high precision.

An analog accelerator has a unique advantage in solving nonlinear equations because it works in continuous time, without steps. As a result, analog accelerators always have up-to-date estimates of nonlinear functions and derivatives. That contrasts with digital, discrete-time systems which must pretend the problem is linear at each step.

In the evaluation of the approach, we discuss tradeoffs concerning analog accuracy and precision, and how we can divide and conquer large problems so subproblems can fit in the analog accelerator. Furthermore, we discuss how to increase the breadth and depth of workloads suitable for analog accelerators. Using the key strengths of an analog model of computing may be one way to deliver performance and efficiency using existing integrated circuit technology in the post-Moore's law era.

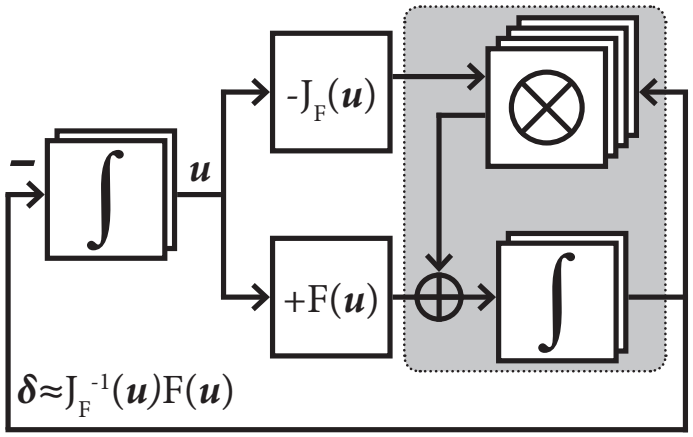

Figure 1 Analog accelerator circuit for solving an ODE, corresponding to a continuous version of Newton's method for solving nonlinear equations.

Problem area: Solving nonlinear partial differential equations (PDEs) are an increasingly important workload as they give natural and accurate models for the physical world. Furthermore, nonlinear PDEs are increasingly useful in systems with limited energy budgets, for applications such as optimal control and fluid dynamics. For example, a mobile robot capable of solving these types of equations would be able to make more informed and optimal decisions navigating the physical world.

We did a thorough survey of nonlinear PDE types, and decided in this paper to focus on solving the canonical 2D viscous Burgers' equation. We chose the Burgers' equation because it is a core part of the important Navier-Stokes equations for fluid dynamics. The Burgers' equation has a single parameter which controls the character of the PDE, and serves to control how nonlinear the problems are in our experiments.

We also studied PDE solution methods, and concluded most solvers distilled the problem to solving nonlinear systems of algebraic equations. Our workload characterization of nonlinear PDE solvers confirms solving nonlinear algebraic equations is the dominant kernel. Therefore, we looked for a way to solve nonlinear algebraic equations in a hybrid analog-digital solver system, which would support many nonlinear PDE solvers while needing little reprogramming.

How it works: To solve nonlinear equations in the analog accelerator, we studied how modern digital computers solve nonlinear equations using algorithms. The prototypical digital numerical method for nonlinear equations is the Newton method,

[1] Y. Huang, N. Guo, M. Seok, Y. Tsividis, K. Mandli, and S. Sethumadhavan, "Hybrid analog-digital solution of nonlinear partial differential equations," in Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50 '17), pp. 665-678, 2017.

[2] Y. Huang, N. Guo, M. Seok, Y. Tsividis, and S. Sethumadhavan, "Evaluation of an analog accelerator for linear algebra," in Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16), pp. 570-582, 2016.

which is an iterative method which makes successive guesses at the solution vector with decreasing error until it converges.

In practice, the Newton method in a digital computer does not always give a correct result, and needs fine tuning of the algorithm in two aspects. First, the choice of the iterative method's step size for updating the guess is important, as the algorithm needs to often reevaluate the nonlinear function and its derivative. Second, the initial guess to the algorithm needs to be close to the correct solution or else the algorithm does not converge.

Doing the Newton method in continuous-time on an analog accelerator has unique advantages. First, the algorithm always has an up-to-date evaluation of the nonlinear function and its derivative, and since the algorithm runs in continuous time, there is no problem in selecting a step size. Second, we show in the paper an improvement to the basic Newton's method called homotopy continuation which allows the analog accelerator to select initial guesses more easily.

Concretely, we do the Newton method in continuous time in an analog accelerator by shrinking the algorithm step sizes until it is an ordinary differential equation (ODE). The analog accelerator solves this ODE, using a circuit setup shown in Figure 1. Integrators at the left side of the diagram store the guess of the solution vector. The integrators feed the guesses to analog subcircuits that multiply and sum the values, to evaluate the nonlinear function and its derivative. Then, the analog accelerator solves a linear algebra equation to approximate the correction term according to the Newton method. The integrators take the correction terms as inputs to update the present guess. To use the analog solver, we charge the integrators to an initial guess. Then, we release the integrators. The analog accelerator solves the Newton method ODE in continuous time until the integrator values are steady, at which point analog-to-digital converters read out the solution.

Hybrid analog-digital system: Once we set up a way to give approximate solutions to nonlinear systems of equations using an analog accelerator, we had to evaluate its usefulness in workloads that a conventional digital computer would handle. The requirements for a hybrid analog-digital approach include high accuracy and precision in the solution and the ability to handle large problem sizes.

To get **high accuracy solutions**, we use the analog accelerator in an analog-digital solver system where approximate and low-precision analog solutions are good initial seeds for a digital solver. This scheme is fruitful because a naïve Newton method solver with a poor initial guess spends most of its iterations trying to find the general area of the correct solution. The results in our paper confirm a digital solver incurs higher time costs as a problem becomes more nonlinear due to this initial search phase. On the other hand, an analog approximate solution allows the hybrid system to fast forward through this phase. Once the hybrid system is in the general area of the solution, the digital solver quickly refines the solution to high precision. The result is the hybrid system solves increasingly nonlinear problems with no significant increase in solution time.

To handle **large problem sizes**, the digital solver divides nonlinear PDE problems into nonlinear systems of equations problems that can fit in the analog accelerator. Since the analog accelerator focuses only on solving nonlinear systems of equations, the existing PDE space and time discretization techniques stay the same, reducing the amount of reprogramming needed to use analog acceleration. If the nonlinear systems of equations are still too big to fit in the analog accelerator, we use red-black nonlinear Gauss-Seidel to further split the problems.

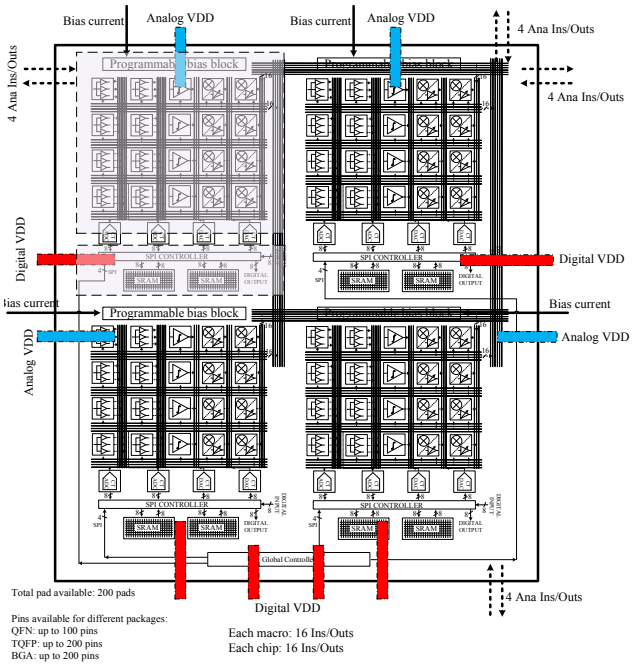

Figure 2 Schematic of a prototyped 4mm×4mm tiled analog accelerator with enhanced calibration for analog components, and designed for multi-chip scalability.

In our paper, we make projections on the performance, power, and area of a tiled-out analog accelerator, up to the point where the analog accelerator size matches that of the largest commercial digital chips. Our comparisons and results assume that as the largest possible analog accelerator. We speculate that given the low power consumption and signal fault tolerance of analog chips, it is possible to build larger analog chips and stack them in ways that are impossible with digital chips.

Physical prototype implementation: We tested our ideas on a two-chip system of physically prototyped analog accelerators. The chips are tiled versions of the programmable microarchitecture from our group's prior work[3], with microarchitecture changes to improve calibration and allow multi-chip scalability. The connectivity for analog signals between chips and between tiles is sparse to match the sparse connectivity of PDEs. Within each tile, the connectivity between analog components is all-to-all to create a variety of nonlinear polynomial functions and their derivatives, giving support for different nonlinear PDEs.

The two-chip system allowed us to test 2D Burgers' equations on a 2×2 grid. We then extrapolated the solution times for analog chips capable of solving larger problems. When an analog accelerator chip capable of solving 16×16 2D Burgers' equations generates approximate solutions to help a GPU running the Newton method, the solution time of the GPU decreases by 5.7× and the energy consumption decreases by 11.6×. These savings are significant since they are the innermost and most intensive kernels of nonlinear PDE solvers.

[3] N. Guo et al., "Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time," in IEEE Journal of Solid-State Circuits, vol. 51, no. 7, pp. 1514-1524, July 2016.

Long-term impact

Nonlinear equations are increasingly important: Scientific computing workloads increasingly rely on nonlinear equations to accurately model the real world. A recent informal survey of applied math literature[4] found Newton methods for nonlinear equations to be the most mentioned algorithm, surprisingly topping other numerical stalwarts like matrix factorization, eigenanalysis, Monte-Carlo methods, and FFT. For comparison, an earlier ranking of algorithms from the turn of the century[5] did not mention nonlinear problems at all.

The importance of Newton methods may surprise computer architects, since scientific computing workload profiles show sparse linear algebra is by far the most important kernel. Accordingly, we tune architectures such as GPUs for linear algebra. The reality is scientific computing workloads oftentimes call linear algebra subroutines from nonlinear equation solvers. Therefore, improvements for nonlinear solvers would reduce the number of calls to linear algebra solvers altogether. Unfortunately, the software behavior of nonlinear solvers is less regular, making it difficult to devise conventional accelerators for nonlinear problems.

Nonlinear is analog killer app: This paper shows analog acceleration has unique advantages in tackling nonlinear problems. That is because the analog accelerator works in continuous time, so that nonlinear functions and derivatives are continuously reevaluated. In comparison, discrete time digital computers must pretend the problem is linear at each time step. If this linear assumption causes problems, the digital computer must invest more iterations and computation time until the linear approximation is good enough. Using an analog accelerator to solve the same nonlinear problem sidesteps these problems because the nonlinear behavior of the analog circuit better matches the nonlinear problem description.

How to do more problems types in analog accelerators: This paper paves the way to finding more problems for the analog accelerator. We do so by converting iterative numerical methods into ODEs, which we then solve in the analog accelerator. In our work, we give three examples of continuous ODEs that solve numerical problems: continuous gradient descent for linear algebra, along with continuous Newton's method and homotopy continuation, both for nonlinear algebra. These examples give us clues on how to find more problems for analog accelerators in the future. For example, iterative numerical methods for important problems such as eigenanalysis and linear programming all have continuous time versions.

Doing iterative numerical methods in an analog accelerator has three advantages. First, analog accelerators work nicely in hybrid analog-digital architectures for iterative methods by giving cheap approximate solutions which a conventional digital computer can then refine. That is possible because iterative numerical methods all work by giving progressively more correct guesses for the problem solution. Second, analog accelerators work in continuous time, without discrete steps, avoiding the choice of step sizes, which control how fast iterative numerical methods updates solution guesses. The choice of these step sizes is often difficult and needs fine tuning. Third, when we use analog accelerators to solve iterative numerical methods, the solution output of the analog accelerator is the final, converged output. Because the output is steady, we can sample the solution with high precision, making it easier to connect the analog accelerator with a digital computer.

How to do more work in an analog accelerator: This paper shows how we can make the analog accelerator do more work, so the ratio of analog computation vs. analog / digital conversion is higher, making analog acceleration more worthwhile. The trick is to have an analog accelerator equivalent of inner loops. For example, we invoked an analog inner loop for linear algebra inside the analog Newton method solver. The Newton method solver is itself an inner loop for homotopy continuation. We implement these inner loops by building subcircuits which converge faster than the overall circuit. Using this trick, we can nest other types of iterative numerical methods, in the same way digital algorithms compose different subroutines.

Broader view: As we enter the post-Moore's law era of computing, unconventional architectures will offer specialized models of computation that uniquely support specific problem types. Two prominent examples are using deep neural networks to support pattern recognition, and using quantum computers for factorization. In our paper we show another specialized, unconventional architecture is to use analog accelerators to solve nonlinear problems. As recent computer architecture conference programs show, these unconventional architectures are now commercially relevant.

Computer architecture researchers will discover other important models of computation in the future. This paper is an example of the discovery process, implementation, and evaluation of how an unconventional architecture supports a specialized workload.

Citation for Test of Time award in 10 years?

This paper finds analog acceleration as the missing hardware primitive for tackling nonlinear mathematical problems. This paper is a first example of composing analog algorithms, allowing researchers to deepen and broaden analog accelerator workloads. With these findings, we can use analog accelerators as domain-specific accelerators, and not just for one-off applications.

[4] https://nickhigham.wordpress.com/2016/03/29/the-top-10-algorithms-in-applied-mathematics/

[5] B.A. Cipra, "The Best of the 20th Century: Editors Name Top 10 Algorithms," in SIAM News, vol. 33, no. 4, May 2000.