A Case for Hybrid Discrete-Continuous Architectures

Simha Sethumadhavan, *Member, IEEE,* Ryan Roberts, and Yannis Tsividis, *Fellow, IEEE*
Columbia University
Email:simha@cs.columbia.edu

Abstract—Current technology trends indicate that power- and energy-efficiency will limit chip throughput in the future. Current solutions to these problems, either in the way of programmable or fixed-function digital accelerators will soon reach their limits as microarchitectural overheads are successively trimmed. A significant departure from current computing methods is required to carry forward computing advances beyond digital accelerators. In this paper we describe how the energy-efficiency of a large class of problems can be improved by employing a hybrid of the discrete and continuous models of computation instead of the ubiquitous, traditional discrete model of computation. We present preliminary analysis of domains and benchmarks that can be accelerated with the new model. Analysis shows that machine learning, physics and up to one-third of SPEC, RMS and Berkeley suite of applications can be accelerated with the new hybrid model.

Index Terms—Hybrid systems, Processor architectures, Design studies,

1 Introduction

DIGITAL accelerators are commonplace in computer systems today either in the form of application-specific accelerators, *e.g.,* XML, Regular Expressions [6], or domain-specific accelerators such as cryptographic or physics accelerators [16]. These accelerators are seen as a way to improve performance and energy-efficiency of computing systems in the face of slowing VLSI scaling [5]. In this paper, we investigate innovations which can increase computational performance and energy efficiency in future systems over and above those offered by digital accelerators. To illustrate the limitations of current accelerator paradigms and make a case for a new model we start with a broad "first principles" characterization of computing (Section 2). We then discuss an alternate computing model, a hybrid of discrete and continuous computing models (HDCA), (Section 3) and its benefits (Section 4). Our analyses indicate that system designers may benefit from designing computational accelerator interfaces that can accommodate both discrete and continuous accelerators and methods for switching seamlessly between the two models. We propose returning to the classical hybrid computing paradigm [10] in the context of modern technology and applications, perhaps with a new hardware/software partitioning that fits current computing requirements and capabilities.

2 Beyond Digital Accelerators

Computing can be explained as a three step process (Fig 1A). First we devise an algorithm for solving a problem, then implement the algorithm, and execute the implementation on inputs to obtain outputs. In the digital computing paradigm (Fig 1B), digital algorithms are devised and executed on devices that support the digital abstraction. If the problem inputs are continuous quantities such as distance, pressure etc., the inputs are approximated (discretized) to their closest digital values which can result in approximate outputs. In a typical digital computing setup, the original algorithm is taken through several intermediate steps or transformations through compilers, execution environments, microarchitecture etc., which introduce or remove overheads.

Application- and domain-specific accelerators are aimed at trimming overheads during the execution step at the microarchitecture level (Fig. 1C). Accelerators implement digital algorithms as application-specific digital circuits thereby avoiding intermediate translation and bookkeeping steps. For example, instead of emulating a cryptographic algorithm with millions of basic operations such as adds and multiplies, a cryptographic accelerator executes it using a one or few instructions [7]. Reducing the number of instructions and the associated bookkeeping reduces energy consumption and improves efficiency. Looking forward, however, it is unclear where further energy efficiency improvements will come from in the accelerator computing paradigm. Relatively incremental benefits are possible by trimming other parts of the computing stack (such as overheads introduced during compilation, by refactoring commonalities among accelerator implementations or virtualization). But once these overheads are removed, what remains is just the computation itself, and the only way to optimize further is to change the algorithmic formulation or even better the computation model[1] underlying the algorithmic formulation.

3 Hybrid Discrete-Continuous Architectures (HDCA)

A fundamental source of inefficiency in computing today is that many real-world continuous phenomenon[2] are solved using the digital computing model *i.e.,* using digital numerical algorithms on digital hardware and with discretized inputs. We attempt to rectify this by providing accelerators that will allow continuous functions to be directly evaluated in hardware (Fig. 1D).

To see the benefits of this approach consider the task of solving a differential equation with known initial conditions to obtain a multivariate function. Let us further say that this differential equation does not have an analytical solution; a reasonable assumption except for the simplest of differential equations. On a digital computer this differential equation will be cast as a set of numerical difference equation and evaluated through a series of discretized computations. In contrast, in an implementation that supports the continuous model, the differential equation can be directly solved on a continuous functional unit without intermediate discretization steps, thereby improving performance and energy significantly. It is important to note that a discrete accelerator that is optimized for solving numerical differential equations, *e.g.,* through optimized datapath layout or a single complex digital functional unit cannot achieve the same performance as the continuous accelerator because the digital accelerator will still require time-stepping. If the continuous accelerators are implemented using modern VLSI techniques the latency of the single step operation can be much lower than the multicycle latencies incurred with digital accelerators. In prior work, speedups of about two orders of magnitude have been reported between continuous and digital solutions for ODEs. [3].

Not only is the continuous model power- and energy-efficient,

Manuscript submitted: 01-Jul-2011. Manuscript accepted: 08-Aug-2011. Final manuscript received: 24-Aug-2011.

1. A computational model defines possible operations and their costs.
2. Computational functions that operate on continuous input variables.

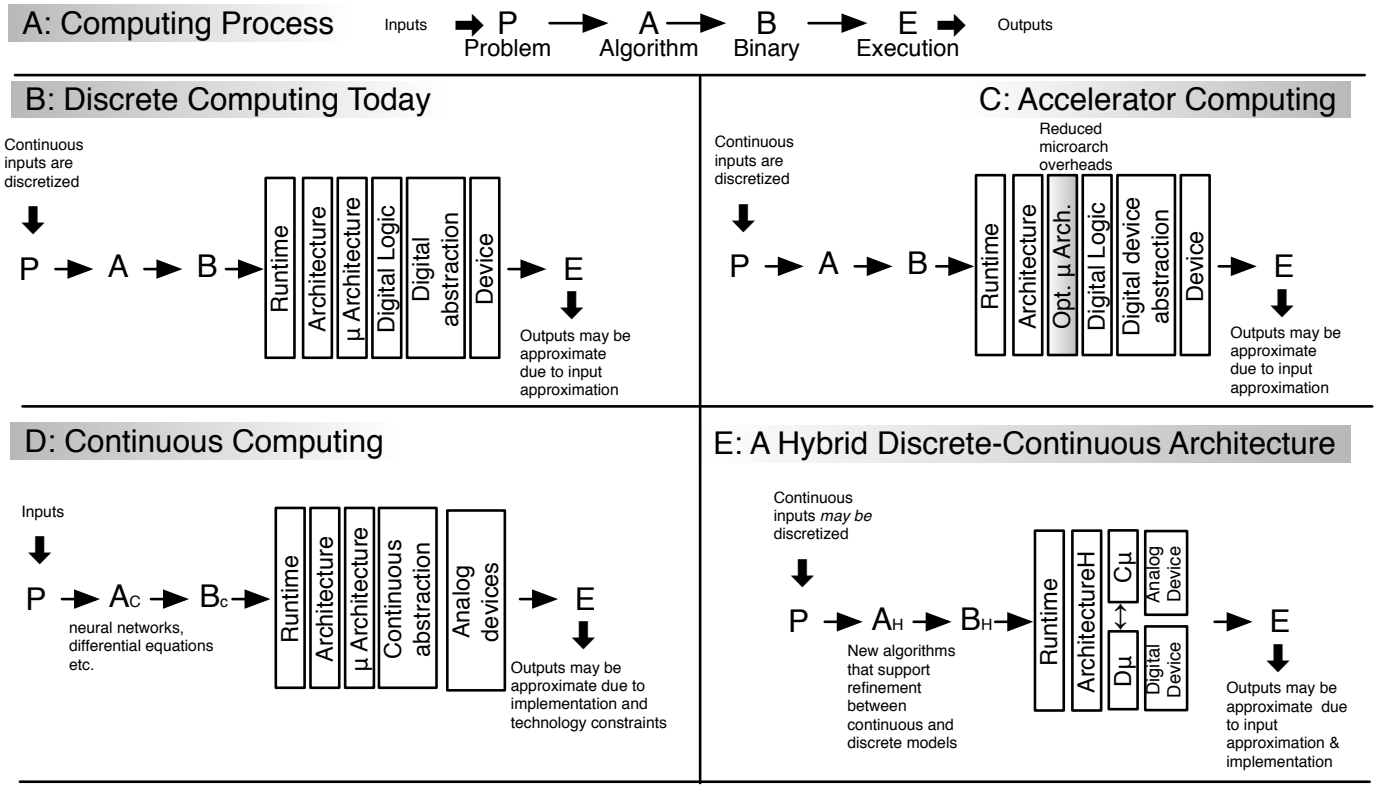

Fig. 1. A first principles illustration of different computing paradigms.

it may also be natural to use for engineering and scientific problems [11], [15]. Going back to the differential equation example in the previous paragraph, in the continuous domain a differential equation need not be cast as discrete numerical difference equations. Avoiding this discretization step is especially beneficial for non-linear systems because, for these systems, choosing the right numerical algorithm requires in-depth knowledge of not only the problem being solved but also all of the subtleties that cause problems with convergence in the discrete domain. Thus, continuous accelerators can improve productivity by bridging the semantic gap between algorithm designers and execution environments.

Despite its benefits, the continuous model cannot stand on its own because of the limitations of the technology that must be used to implement it. While analog circuits can implement continuous functions, reliable continuous storage is not feasible. Further, analog computation tends to be error-prone and non-reproducible. These limitations can be mitigated by combining the digital and analog models (Fig. 1E). It is likely that error rectification and reproducibility can be obtained with digital circuits. Further discrete model may also be used to store and process inputs or sequence through multiple operations. Extending discrete and continuous aspects of the hardware substrate to all levels in the system stack can help algorithm designers to partition their workloads between the discrete and continuous domains, and workaround the limitations of pure analog implementations.

4 What Applications can Benefit from HDCA?

In the 1960s, analog computers experienced a couple of decades of rapid development [14]. Ultimately, with advances in fabrication and the development of design automation tools for digital designs, the discrete computing model won out; but not before a large class of mathematical models were understood in the analog computing framework. Most notable among these are differential equations (linear ordinary differential equations with constant or varying coefficients, nonlinear ordinary differential equations, and partial differential equations), algebraic matrix models (the solution of simultaneous equations, matrix inversion, vector/vector, vector/matrix, and matrix/matrix operations) and the simulation of a complex physical systems. In this section, we analyze how these technologies can be leveraged in the context of HDCA.

4.1 HDCA for Approximate Algorithms

Learning Today, several application-specific analog VLSI systems exist to solve a range of problems in learning algorithms. Most notably, a significant amount of work has been done applying analog VLSI to neural networks [12]. Recently Amant *et al.* used an analog version of the neural predictor to improve the power and performance of branch prediction [1]. Analog accelerators are a good match for neural networks because continuous transistor characteristics can be used to implement the sigmoid function used for computing weights. Recent work has also proposed an analog VLSI implementation of support vector machine learning and classification [13] and pattern classification and sequence estimation [2].

Interactive Applications A second application domain that maps well to HDCA, but has to the best our knowledge received no attention, is interactive applications, where the required precision is set by what an end user can detect. It has been reported that humans cannot visually detect inaccuracies within 3% in direct vision and up to 20% for objects that are in the field of view but not in direct vision [8]. Yeh *et al.* [16] developed a general-purpose physics processor used to accelerate calculations for real-time gaming by utilizing a low-precision digital architecture that relies on the fact that visual inaccuracies cannot be detected. This approach still suffers from using a discrete time stepping algorithm to solve a continuous problem, however. For such problems, the HDCA model with an analog accelerator is ideal since it ultimately reduces to solving a set of differential equations.

4.2 HDCA for High Accuracy Applications

Even when high accuracy is required, many applications can benefit from a analog accelerator coupled with a traditional digital discrete core or accelerator. For example, traditional iterative methods used to solve a large class of problems are computationally very expensive. In many of these applications, convergence (and hence the time to completion) of the algorithm is determined by a variety of factors including: how close the initial seed solution approximates the desired solution, how much work is required to set up successive iterations, how well the algorithm approximates certain problem parameters to move the algorithm to the next iteration, and the error tolerance for the final solution. Next we describe how different facets of iterative applications can be improved with the HDCA model.

Seeding A HDCA implementation can accelerate several critical steps in high-accuracy applications without impacting the final accuracy. For example, a continuous, and imprecise analog accelerator can be used to supply the seed solution to the iterative algorithm to within a few percent significantly reducing the total number of iterations [10], [3]. This is especially beneficial for non-linear problems that may not converge unless they are started sufficiently close to the true solution.

Boosting Similar to seeding, we suggest that intermediate steps in a computation may also be sped up using the HDCA model. For example, in sparse linear systems, a continuous analog accelerator can be used to calculate key steps, such as the preconditioner in the Conjugate Gradient Method to accelerate convergence. Similarly, adaptive grid sizing can be accelerated using analog implementations. Having the continuous accelerator as part of the same die as the discrete core makes it possible to use the analog accelerator for much finer-grained computation. To the best of our knowledge, we are first to suggest the broad applicability of boosting though some prior work hints at the benefits of seeding techniques in the context of earlier analog proposals [9], [4].

4.3 HDCA Coverage for Standard Benchmark Suites

To better understand how current and future ("killer") applications map to the hybrid model, we examined the SPEC CFP2006 benchmark suite, Intel's RMS suite, and the Berkeley Dwarfs suite. Each benchmark and/or application domain was characterized based on the underlying algorithms and computation. For each, we determined if it maps well to the analog domain (*e.g.,* differential equations/algebraic models), if it maps well to analog after some problem transformation of the original problem (*e.g.,* linear programming), or if the problem simply does not map well to the analog domain (*e.g.,* table look-ups). Table 1 highlights our findings.

A number of the examined benchmarks fall under the Dense Linear Algebra and Sparse Linear Algebra categories of the Berkeley Dwarfs: fluid dynamics (410.bwaves), quantum chemistry (416.gamess and 465.tonto) and linear programming (450.soplex) from CFP2006, support vector machines, quadratic programming, and a number of partial differential equation applications from Intel RMS. These applications consist solely of additions and multiplications of a variable by a constant and can be implemented efficiently in the analog domain. More general and efficient implementations that utilizes analog integrators to solve for the unknowns in the given system are also possible. This is in contrast to a digital implementation, which typically requires some sort of numerical algorithm (*e.g.,* LU factorization) to perform the most computationally intensive tasks in these benchmarks.

Spectral methods, such Intel RMS's computational fluid dynamics and cloth simulation, are dominated by additions and multiplications (Fast-Fourier Transform) and also map well to the analog domain. These operations are fast in both the digital and analog domain, so the expected speedup of an analog implementation over a digital implementation may be of little benefit. A significant chip area advantage is expected from the analog implementation, however. For example, assuming 8-bit additions and multiplications, a *naivé* digital implementation would require approximately 240 transistors for addition and 3000 transistors for multiplication. An analog implementation can perform the same operations with far fewer transistors resulting in lesser area and power.

Simulation of dynamic systems, N-body methods, Structured and Unstructured Grids, require solving ordinary differential equations in some cases (*e.g.,* applying the Barnes-Hut algorithm to galaxy simulation), or more often a set of coupled partial-differential equations which clearly map well to the continuous model. The majority of the CFP2006 benchmark suite including: molecular dynamics (435.gromacs, 444.namd), magneto hydrodynamics (434.zeusmp), general relativity (436.cactusADM), fluid dynamics (437.leslie3d, 470.lbm), finite element methods (447.deal1II, 454.calculix), Maxwell's E&M solver (459.GemsFDTD), quantum crystallography (465.tonto), and weather modeling (481.wrf), falls within this realm.

It is less clear how the remaining benchmarks in these three suites and application domains map to the analog domain. Specifically, ray tracing (453.povray) does not appear to be a good fit for analog, and it is unclear if speech recognition is likely to benefit from analog. Similarly, some of the Berkeley Dwarfs: MapReduce, Combination Logic, and Finite State Machines do not lend themselves to the analog computing model. For the remaining Berkeley Dwarfs: Graph Traversal, Dynamic Programming, Back-Track and Branch and Bound, and Graphical Models, the mapping to the analog domain is entirely application-specific. For example, if the intermediate calculations on nodes of the graph traversal require calculations that are amenable to analog computing, then the application maps well to analog; and if they do not, then the application is unlikely to benefit from analog.

While a surprisingly large number of these benchmarks and application domains map well to the analog domain, it is important to note that both accuracy limitations and chip area constraints pose limitations on any analog implementation. For instance, if the problem requires a relative error of less than 1% or requires a large number integrations, then a direct analog implementation that operates on the entire problem set at once may not be practical. In many instances these issues can be alleviated, *e.g.,* splitting a large mesh into a series of sub-meshes and performing calculation on the sub-meshes sequentially; or utilizing a hybrid analog/digital approach to have the analog rapidly calculate the solution at non-critical points in the system and using digital parts to accurately calculate solutions to the reduced "critical" system. Ultimately, the most interesting applications for an analog accelerator will likely fall within the hybrid discrete/continuous framework such as using the analog accelerator solving the equivalent linear program in the Branch-and-Bound method for integer programming, or calculation of particle interactions at each node in the Barnes-Hut algorithm for N-body simulation.

5 Related Work

The idea of combining analog and digital computers was popular in the 1970s [10], [11], [14]. The idea fell out of favor because of superior performance of digital computers. We believe that are two major reasons to reconsider hybrid computing going forward. First, the last decade has seen the emergence of several new applications that can greatly benefit from the HDCA model including games and learning. Second, higher levels of VLSI integration allow analog and digital components to be used in tandem while switching between them at finer granularities.

TABLE 1
Applications from three benchmark suites and their suitability for HDCA.

Maps to Analog		
Benchmark	Suite	Domain/Example
410.bwaves, 437.leslie3d, 470.lbm	CFP2006	Fluid Dynamics
416.gamess, 465.tonto	CFP2006	Quantum Chemistry
Cloth, Face simulation	Intel RMS	PDE
450.soplex	CFP 2006	Linear Programming
Support Vector Machine	Intel RMS	
Quadratic Programming	Intel RMS	
Spectral Methods	Berkeley Dwarfs	PDE/FFT
N-body Methods,Structured & Unstructured Grids	Berkeley Dwarfs	Dynamic Systems simulation, PDE
435.gromacs, 444.namd	CFP 2006	Molecular Dynamics
434.zeusmp	CFP 2006	Magneto Hydrodynamics
436.cactusADM	CFP 2006	General Relativity
454.calculix, 447.deal1II	CFP 2006	Finite Element Methods
459.GemsFDTD	CFP 2006	Maxwell's equations
465.tonto	CFP 2006	Quantum Cystallographgy
481.wrf	CFP 2006	Weather Modeling
Type of Mapping is Problem Dependent		
Benchmark	Suite	Domain/Example
Graph Traversal	Berkeley Dwarfs	Decision Trees/Natural Language Processing
Dynammic Programming	Berkeley Dwarfs	Traveling Salesman
Back-Track and Branch and Bound	Berkeley Dwarfs	Integer Programming/Network Simplex
Graphical Models	Berkeley Dwarfs	Hidden Markov Models/Nueral Networks
No Mapping to Analog		
Benchmark	Suite	Domain/Example
453.povray	CFP 2006	Image Ray-Tracing
482.sphinx3	CFP 2006	Speech Recognition
MapReduce	Berkeley Dwarfs	Monte-Carlo simulation
Combination Logic	Berkeley Dwarfs	Hashing
Finite State Machines	Berkeley Dwarfs	Video Compression/Text processing

6 Conclusions

In this paper we pointed out the inefficiencies in digital accelerators and studied the benefits of combining discrete and continuous computing models. We suggested that an implementation of a continuous model of computation using analog accelerators can improve the energy-efficiency for a number of problems. Our analysis shows that several current and emerging workloads such as physics processing and data mining tasks use differential equations extensively and can be solved efficiently using HDCA accelerators. These domains do not have stringent accuracy requirements, which is important because analog implementations of continuous operations are likely to have low noise immunity and consequently lower accuracy than their pure digital counterparts.

The applicability of continuous accelerators, of course, can go beyond these approximate computing domains. When high accuracy is required, an analog circuit can be used to produce a low precision initial guess that a digital computer can refine to quickly get a high-precision final answer. This computing trick, and other "refinement techniques we have developed, can be applied to about one-third of the benchmarks in the SPEC CFP2006, Intel RMS, and the Berkeley Dwarfs suite. Much exciting works remains to be done in this area, including development of analog microarchitectures, programmable interfaces between analog and digital components, and understanding overheads of switching between discrete and continuous domains.

7 Acknowledgements

Sethumadhavan's research is funded by grants from DARPA, AFRL (FA8750-10-2-0253, FA9950-09-1-0389), the NSF CAREER program, gifts from Microsoft Research and Columbia University, and software donations from Synopsys and Wind River. Roberts conducted this research as a GRA in Sethumadhavan's Lab.

References

[1] R.S. Amant, D.A. Jiménez, and D. Burger. Low-power, high-performance analog neural branch prediction. In *Proceedings of the 2008 41st IEEE/ACM International Symposium on Microarchitecture-Volume 00*, pages 447–458. IEEE Computer Society Washington, DC, USA, 2008.

[2] S. Chakrabartty and G. Cauwenberghs. Sub-microwatt analog VLSI support vector machine for pattern classification and sequence estimation. *Adv in Neural Information Processing Systems*, 17.

[3] G.E.R. Cowan, R.C. Melville, and Y.P. Tsividis. A vlsi analog computer/digital computer accelerator. *Solid-State Circuits, IEEE Journal of*, 41(1):42 – 53, jan. 2006.

[4] C.C. Douglas, J. Mandel, and W.L. Miranker. Fast hybrid solution of algebraic systems. *SIAM J. Sci. Stat. Comput*, 11:1073–1086, 1990.

[5] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, and D. Burger. Dark Silicon and the End of Multicore Scaling. In *Proceedings of the 38th International Symposium on Computer Architecture*, June 2011.

[6] H. Franke, J. Xenidis, C. Basso, B. M. Bass, S. S. Woodward, J. D. Brown, and C. L. Johnson. Introduction to the wire-speed processor and architecture. *IBM Journal of Research and Development*, 54(1):3:1 –3:11, 2010.

[7] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A.Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz. Understanding sources of inefficiency in general-purpose chips. In *ISCA*, pages 37–47, 2010.

[8] Jason Harrison, Ronald A. Rensink, and Michiel van de Panne. Obscuring length changes during animated motion. *ACM Trans. Graph.*, 23(3):569–573, 2004.

[9] WJ Karplus and RA Russell. Increasing digital computer efficiency with the aid of error-correcting analog subroutines. *IEEE Transactions on Computers*, 100(20):831–837, 1971.

[10] G.A. Korn. The impact of hybrid analog-digital techniques on the analog-computer art. *Proceedings of the IRE*, 50(5):1077–1086, May 1962.

[11] G.A. Korn and T.M. Korn. *Electronic analog and hybrid computers*. McGraw-Hill New York, 1972.

[12] Carver Mead. *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[13] S.Y. Peng, B.A. Minch, and P. Hasler. Analog VLSI implementation of support vector machine learning and classification. In *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, pages 860–863, 2008.

[14] A.I. Rubin and J.B. Mawson. Hybrid computation 1976 and its future. *Computer*, 9(7):37 –46, july 1976.

[15] J. F. Traub. A continuous model of computation. *Physics Today*, page 39, 1999.

[16] Thomas Y. Yeh, Petros Faloutsos, Sanjay J. Patel, and Glenn Reinman. Parallax: an architecture for real-time physics. *SIGARCH Comput. Archit. News*, 35(2):232–243, 2007.