Opportunities in Physical Computing driven by Analog Realization

Jennifer Hasler
Georgia Institute of Technology, Atlanta, GA 30332–250 USA E-mail:jennifer.hasler@ece.gatech.edu

Abstract—**In the past, discussions on the capability of analog or physical computing were only of theoretical interest. Digital computation's 80 year history starts from the Turings original model of computation to ubiquitous modern computational devices. The modern development of analog computation started with almost zero computational framework. Today, we have significant programmable and configurable physical computing systems. The focus of this paper is to have these discussions given the very real potential of ultra-low power physical computing systems. This work considers the current state of analog computation, energy efficient computation, and analog numerical analysis, moving towards starting a unified analog-computing framework, including quantum computing, as part of physical computing.**

The first wave of Neural Network (NN) research was central to my early graduate student days (1987-1992). Those days were filled with speculation about how the brain computed, about physical[1] computation, its relationship to digital computation, and if physical computation could exceed projected digital computation solutions. Modern analog computation started together with Neuromorphic (including NN) revitalization in the 1980s (e.g. [1], [2], [3]); these two fields have been tightly linked. Further, was there a question about the existence of an analog Turing machine, and if so, what was its theoretical and practical capability. In those days, Moore's law [4], [5] was a given, just like gravity; one would have to exceed digital's perceived solutions to be competitive when a new technology would be available.

Many of these discussions happened over beverages later in the day. We would have equally strong proponents for and against each position. Discussions would rage on about Hopfield's work solving the Traveling Salesman Problem (TSP) [6], and whether these results could eventually show NP class problems could be solved in polynomial (P) time by physical computing. Although theoretically interesting, these questions seemed to have little relevance to any practical computing system. In the end, the discussions were left at that establishment. Almost no one believed serious analog computing systems would be realized.

Coming many years later to today, we have significant programmable and configurable physical computing systems (e.g. [7]). The discussions are no longer simply theoretical, but a key building block towards unlocking the potential of physical computation. These forgotten conversations must be resumed. The focus of this paper is to have these discussions given the very real potential of ultra-low power physical computing systems. We firmly believe these discussions are just beginning.

[1]we use physical and analog computing interchangeably to remove any bias that analog computing means linear computing

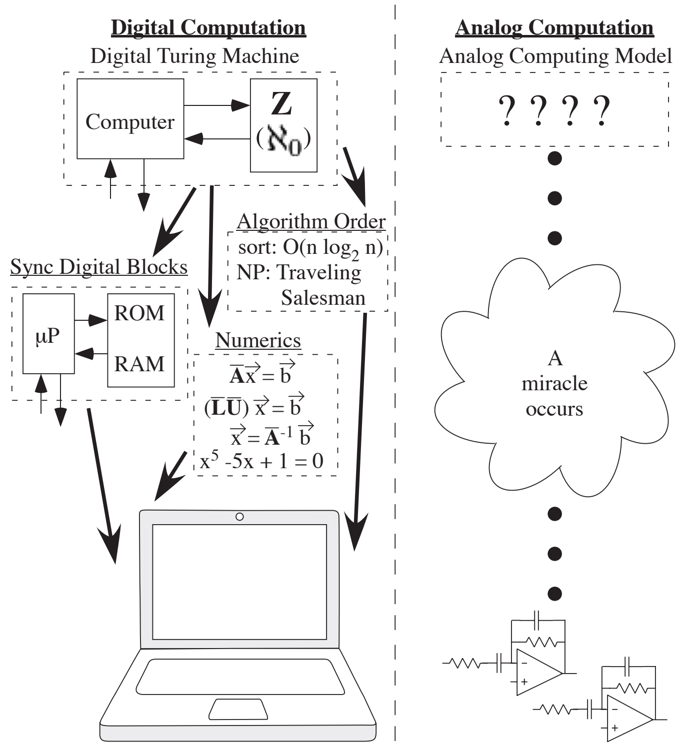


Fig. 1. Digital and Analog Computation approaches based on their fundamental framework (or lack of it). Digital Computation builds from the framework of Turing Machines, setting up capability of computer architectures, computer algorithms, and resulting numerical analysis. This framework becomes the basis for our day to day digital computing, such as laptop computing. Analog Computation is perceived to have little computational modeling, as well as architectures and algorithms. The resulting analog computing designs, where built, seems more like bottom-up artwork rather than top-down digital computing design.

I. Overview of Traditional Digital and Analog Computing Perspectives

Figure 1 illustrates the traditional viewpoint of digital and analog computation. Digital computation 80 year history starts from the Turings original model of computation [8], a model based upon bookkeeping businesses at the time. The model (Fig. 1) requires countable alphabets for inputs, outputs, and the resulting memory *tape* processed through a single machine. He proved that Turing machines would be capable of performing any conceivable mathematical computation if it was representable as an algorithm.

978-1-5090-1370-8/16/$31.00 ©2016 IEEE

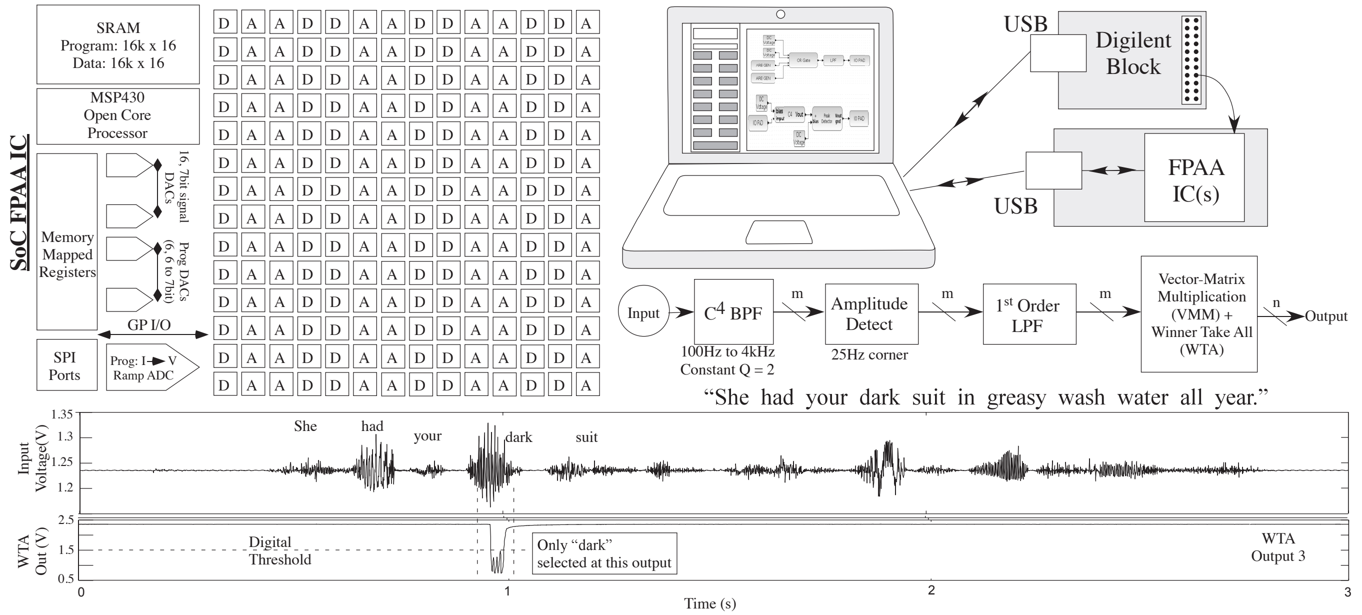


Fig. 2. The SoC large-scale Field Programmable Analog Array (FPAA) device showing a command-word speech recognition. We show the high-level block diagram of the SoC FPAA device (left), a typical measurement setup and computational block diagram for command-word speech recognition, and measured input and classifier output response classifying the word *dark* in the TIMIT database phrase. This analog computation ($< 30\mu$W) is radically different than the class of expected analog operations.

Digital computing became ubiquitous because of inexpensive digital electronics directly a consequence of Moores law scaling [4]. Or as Moore would reflect later

> "So I took that first few points, up to 60 components on a chip in 1965 and blindly extrapolated for about 10 years and said okay, in 1975 well have about 60 thousand components on a chip. Now what was I trying to do was to get across the idea that this was the way electronics was going to become cheap." – Gordon Moore, 2005 [9].

This prediction [4], and an updated prediction in 1975 [5], continued for decades. Proportionally shrinking transistor dimensions gets nearly the same device with quadratic decrease in parasitics and quadratic increase in its computational energy efficiency [10], [11]. The large number of transistors would transform digital computation through the VLSI concept, effectively invented and evangelized by Carver Mead and Lynn Conway [12]. Digital computation empowering whole communities to program digital systems for a wide range of applications (e.g. microprocessors (μP)), communities that would not do physical digital design. These developments eventually lead to further stratification, including development of standard cells, verilog digital representation, FPGAs, and whole ranges of software developments. A roadmap of future directions typically arrived as new technologies were available.

The perceived situation for analog computation could not be more different (illustrated in Fig. 1). Analog computation seems to be a bottom-up design approach practiced by a few artistic masters. One would be hard pressed to find someone with knowledge of analog computing theory other than using a combination of passive components (e.g. resistors, capacitors) around an op-amp device. Certainly there are no textbooks explaining the analog equivalent to a Turing Machine or analog system synthesis. Many might believe analog would be more efficient, but unlikely how to quantify that improvement.

Analog computation has an old history going back to mechanical differential analyzers for solving ODEs [13]. The computation quantity was represented by a physical measure, such as water in pipes or electronic circuits. Both mechanical and electrical physical computing systems were used for a century. Traditional analog computing was considered by multiple authors (e.g. [14], [15]); the solutions were a series of special case solutions with little overarching computational model. The General Purpose Analogue Computer (GPAC) was one of the few theoretical analog models (proposed by Shannon [16]) equating analog computation as a differential analyzer. GPAC is a set of four basic operations (some nonlinear) boxes, connected through constrained input and output rules. Although the model could eventually represent differentially algebraic functions [17], [18], the model was restrictive in the sense that it did not correspond to obvious physical devices that could eventually be inexpensively constructed. A few reviews of early Analog computing can be found (e.g. [19]). Only recently with the physical reality of significant physical computing approaches [7] have design methodologies and abstractions based on highly repeatable devices emerged and are embedded in design tools [20], [21].

The modern development of analog computation (starting 1980s) started with almost zero computational framework. As individuals started looking at physical computing systems, either from inspiration of neurobiology or from elegant circuits built in CMOS ICs, they developed without any guidance, as well as any bias, of previous models (e.g. [3]). Analog

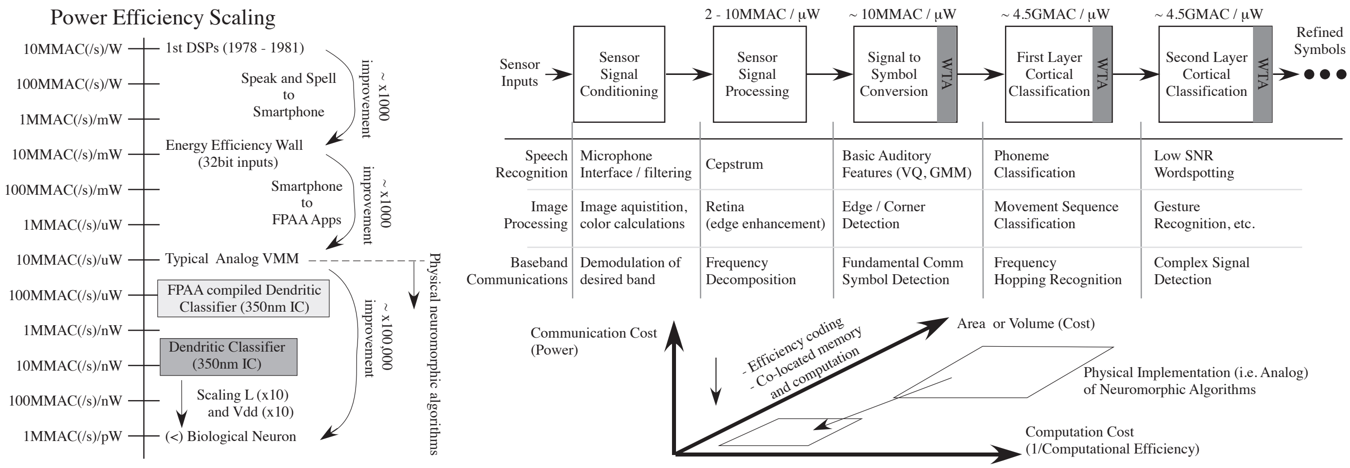


Fig. 3. Comparison of power efficient computational techniques in MAC (/s) / W, including digital, analog Signal Processing (SP) techniques, and the potential for neuromorphic physical algorithms. Three orders of magnitude has produced amazing improvements in digital technology from speak-and-spell devices [59] to current day smart phones. Three orders of magnitude in analog SP approaches has the promise of similar advancements as it becomes a stable capability. Biological neurons show a potential of five more orders of magnitude of improvement, opening further opportunity for efficient computational devices. We also show a typical signal processing chain using configurable analog approaches and neural based classifiers. Once the input signal becomes established as a refined probability of low-level symbols, through a WTA approach [36], we have a cascade of classifier layers typical of processing in cortex. Finally, all the three dimensions (computational efficiency, communication power, and system area) are essential to optimize to the energy, complexity, and area constraints of large-scale neuromorphic systems. Using physical based (i.e. analog) approaches help to decrease computational efficiency and system area, and heavy use of local communication, integration of memory and computation, as well as low-event architecture reduces the communication power required.

computing had to build its entire framework. Mead's 1990 paper hypothesized that analog computation, in particular multiplication, would be at least x1000 greater computational energy efficiency than custom digital solutions [22]. Chawla, et. al (2004) would later experimentally prove this hypothesis for Multiply-ACumulate (MAC) operations between analog and digital approaches [23]. And early efforts would start to consider models of analog computing, beginning to uncover analog computation could be more powerful than digital Turing Machine model [6], [24], [25], [26].

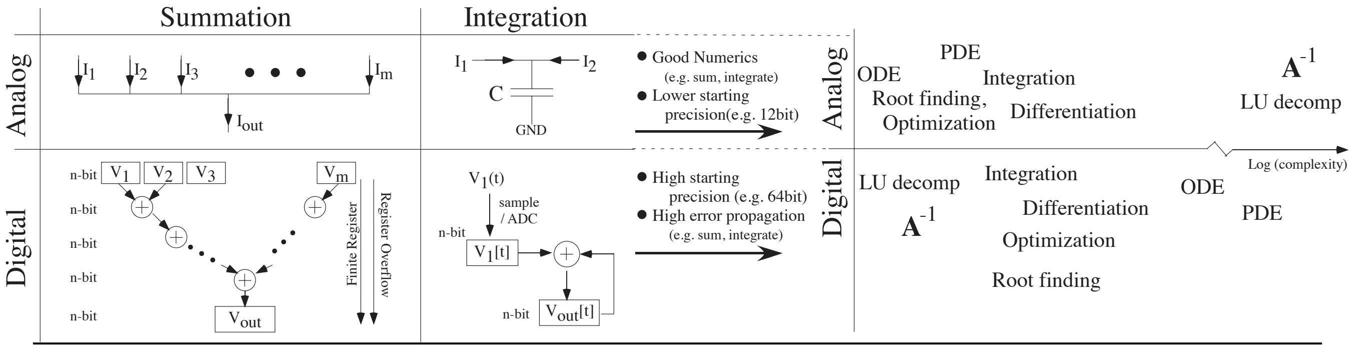
Analog computation did not initially have a memory device. The first 8-10 years of analog NN development struggled due to the lack of a memory element. The Single Transistor Learning Synapse (STLS, 1995) finally gave analog CMOS computation a long-term memory element that could be embedded into the computation (and adaptation) [27], [28], [29]. This structure demonstrated the first crossbar computational model, currently popular with novel nanodevice research.

II. Analog Computation System: SoC FPAA IC for Signal Processing

Analog Computing has grown up. Analog computing is programmable and configurable, through a range of large-scale Field Programmable Analog Arrays (FPAA). These configurable devices compare favorably against custom designs; unlike FPGA designs, FPAA architectures are open to the academic community. Floating-Gate (FG) based FPAA designs, based on STLS analog memories, enable considerable parameter density; memory and computation capability are closely linked in analog computation. FPAA devices enable both analog and digital computation[7], while retaining the x1000 improvement (as predicted by [22]) in computational energy efficiency compared to custom digital solutions (e.g. [31]).

Analog computing is different from emulated ODEs through Op-amps and passive components and is different from front-end sensor preconditioning before a data converter. Figure 2 shows an auditory classifier system demonstrated in the SoC FPAA (< 30 μW power consumption, 350nm IC) [7]. The circuit components involve transconductance amplifiers and transistors (and similar components) with current sources programmable over six orders of magnitude in current (and therefore time constant) [30]. All devices, including crossbar routing, is utilized for potential computation [32]. The entire application was developed in high-level tools implemented in Scilab / Xcos and compiled to working FPAA hardware [21]. A compiled analog acoustic command-word classifier on the FPAA SoC requires x1000 lower power than digital solutions to experimentally recognize the word dark in a TIMIT database phrase. The authors expect future system optimization in the same SoC FPAA. Recently, the SoC FPAA device demonstrated the capability to learn classifier parameters [33], [34] in addition to the original classification capability, enabling this approach towards embedded machine learning applications. The novel classifier structure [35], used in the SoC FPAA demonstration [7], utilizes one layer of a Vector-Matrix Multiplication (VMM) [23], [31] and a Winner-Take-All (WTA) [36] computation. This VMM +WTA classifier is experimentally demonstrated to be a universal approximator, firmly destroying Minsky's early issues with even one-layer neural network structures [37].

Analog computing is built on a wide demonstration of programmable and configurable approaches. The programmable FG circuits enable high-matching analog circuits [38], includ-



	Digital	Analog
Precision	High starting precision, low cost adding more	lower starting precision (e.g. 10, 12, 14 bit)
Summation Noise	High numerical noise accumulation (summation)	ideal summation
Latency	computational latency (need for pipelining)	minimal latency
Computational Efficiency (VMM)	10MMAC (/s) / mW (32 bit)	10MMAC (/s) / μW (x1000, 12bit)
Intellectual Tradition	Long numerical tradition	a few artistic experts, sometimes agree

Fig. 4. Illustration of basic computing operations, summation and integration, between analog and digital approaches. Analog approaches tend to be nearly ideal in these situations, where digital approaches accumulate significant noise and headroom errors in these processes. Analog approaches need to be aware how these operations interface with the rest of the computation circuitry. Finally, we summarize the computation between digital and analog approaches.

ing references [40], amplifiers [39], sensor interfaces [41], filters [42], [43], and data-converters [44], [45], [46], enabling dense high SNR devices. These programmable circuits enable power-efficient (x1000 versus custom digital) signal processing demonstrations [47] using filterbanks [48], [42], Guassian Mixture Models (GMM) [49], Support Vector Machines [50], VMM + k-Winner-Take-All (WTA) classifiers [35], and adaptive filters [51], towards accoustic [52] and imaging [53], [54] applications. Configurable (e.g. FPAA) devices retain the factor of x1000 improvement in power efficiency for signal processing functions [31] on an integrated mixed-mode (analog+digital) fabric [55], [7] for applications already mentioned as well as image processing [56], classifiers [35], robotics and pathplanning [57], [58].

III. Energy Efficient Computation: Digital, Analog, and Neuromorphic Approaches

Figure 3 shows a spectrum showing the computational efficiency of various technologies, including digital computing, analog computing, as well as best estimate of biological neuron computation. The potential of 8-9 orders of magnitude of computational efficiency improvement illustrates we are just at the beginning of computational efficiency scaling, not at the end as predicted by the end of Moores law [9] or the digital energy efficiency wall [60]. Efficient neuromorphic systems could be defined as those physically implemented algorithms that improve power efficiency beyond the analog SP metrics.

Previously we showed that building Si models of human cortical processing was possible using current CMOS technology [61], providing a roadmap for building cortical structures. Even with this roadmap, the community has much to understand in terms of the dynamics of neural computation. Neurobiological computation is one of the best examples of computationally efficient analog / mixed signal computing [61], [62]; these systems are energy constrained, and therefore communication constrained [63].

Modeling of neurobiological systems based on fundamental Si models of channels [64] forms one area of dynamical system modeling. These systems utilize biophysical connections between biological channels and silicon MOSFET transistor channels [64], single-transistor synapses [27], [28], synapse learning (e.g. STDP) [65], dendrites [66], and biological networks [67], [68], [69]. We have zero gap between neurobiological modeling and Si hardware implementation that can move towards applications. The demonstrated engineering application of these networks includes use of two-dimensional grids for path planning using energy surface [70] and active neuron approaches [71], dendritic modeling [66] for wordspotting computation [69], and retina-like image processing.

IV. Analog Numerical Analysis

After one appreciates the very real and practical possibility of programmable and configurable analog systems, the next questions concern the noisy or low-precision issues real and perceived in analog systems. The issue rightly starts noting mismatch between typical analog components. Programmability is essential to addressing this issue. The use of FG devices enables directly programming out these issues, including accounting for a range of temperatures. Neurobiological systems seem to adapt around its mismatches to create precision in its analog computational structures as well. Without this level of programmability, large-scale analog computation is nearly impossible.

The question then turns to a question of the apparent low SNR of individual analog computations aggregating into a larger computation. The question is a digitally centric viewpoint requiring more discussion to fully appreciate this subject. In the end, we find that digital systems have relatively

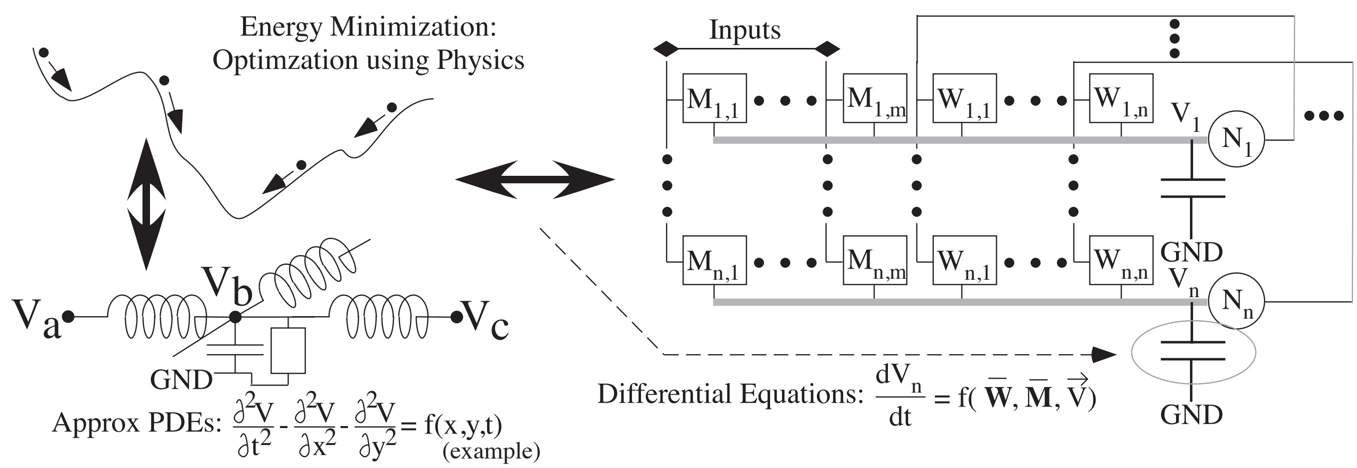


Fig. 5. Most physical systems are modeled as coupled ODEs or PDEs typically modeling a particular potential surface (e.g. Energy) to be minimized. These systems tend to be recurrent ODE systems or state-holding PDE systems (e.g. hyperbolic). One starts these systems at an initial condition and the system settles to a steady-state solution. Hopfield Networks and ARNN fit into this description.

inexpensive high resolution (16, 32, or 64bit) but with noisy numerics. Analog systems have higher cost for starting resolution (8 to 12bit is typically reasonable) but with far less noisy numerical calculations. Digital precision cost is polynomial in the number of bits ($\log_2$(precision)), where analog precision cost is polynomial in precision. At low precision (8 to 10bits or less), analog precision is less expensive than digital due to the reduced overhead.

For the most part, the field of analog numerical analysis was never developed. These techniques provide understanding output SNR of digital or analog computation. As digital computation became more powerful and less expensive in the 1970s and 1980s, digital numerical analysis techniques were already an established and growing discipline. These techniques provided potential computational roadmaps for the exponential computational increase from the digital VLSI revolution [12]. The corresponding analog numerical analysis needs to be developed (and pulled out of the lore of the few master research groups) for corresponding growth in analog computation. We summarize these concepts in this paper; a full treatment is beyond the scope of this discussion.

Figure 4 shows a comparison of summation and integration operations in digital and analog computation. Analog summation by charge or current (change of charge with time) is ideal due to KCL, the physical summation of carriers. Depending on how the designer uses this output current in further calculations can result in nonlinear effects; issues arise from the capability of the analog algorithm design. Analog integration is also ideal, typically performed as a current (or sum of currents, I_k) on a capacitor (of size C) as

$$C \frac{dV_{out}}{dt} = \sum_{k=1}^{n} I_k \quad (1)$$

where V_{out} is the computation result. Analog naturally has infinitesimal timesteps, with no errors due to these timesteps, eliminating accuracy issues arising from the order of numerical integration approximation. This framework shows why analog computation is ideally suited towards solutions of ODEs.

Digital summation is filled with noise errors. The sum of two n-bit numbers half the time will be an n+1 bit number. One either handles fixed-point arithmetic issues by having large enough total resolution for the summation to avoid overflow nonlinearities or handles floating-point arithmetic issues by having enough mantissa bits to account for the LSB noise source. The issue for digital computation is a large number of aggregate summations; 1024 summations will lose 5 bits of precision on average and 10bits worst case. For 16bit registers, this error can be significant; we have not addressed any further errors due to catastrophic subtraction of similar numbers. Integration, often implemented as a sequence of summations, further compounds these numerical issues. Further, integration must be approximated by a set of small regions. Too few steps, and one gets low accuracy. Too many steps, and the summation errors due results in low accuracy. The ODE solution further complicates these issues with numerical stability issues, order of derivative approximations, as well as stiff ODE computations. An engineer faced with these issues most naturally would try to reformulate a problem (ODE $\rightarrow$ Linear equation solution) to avoid these issues where possible.

The algorithm tradeoff between analog and digital computation directly leads to the tradeoff between high-precision with poor numerics of digital computation verses the good numerics with lower precision of analog computation. Digital computation focuses on problems with limited number of iterations that can embody high precision (e.g. 64 bit double precision), like LU decomposition (and matrix inversion). The LINPACK metric[2][72] makes complete sense to evaluate computing engines when the fundamental computing operations are LU decomposition. Classical digital numerical analysis courses begin with LU decomposition and move to significantly harder computations in optimization, ODE solutions and PDE solutions; many engineering problems try

[2]LINPACK is a benchmark measure how fast a digital computer solves a dense n by n system of linear equations Ax = b

to move computation towards matrix inversion and away from ODE solutions.

Analog computation solves difficult numerical applications that are tolerant of lower starting precision for computation, such as ODEs and PDEs. Simple operations like VMM is fairly similar in tradeoffs between analog and digital approaches, particularly when using real world sensor data starting off with lower precision (e.g. acoustic microphones at 60dB, CMOS imaging at 50-60dB, etc.). Many ODE and PDE systems have correlates in other physical systems found in nature that are the focus of high performance computing. The resulting time / area efficiencies for analog computation model a physical system by directly *being* the system to solve. This high-speed computation enables low-latency signal processing and control loops. One would want to avoid analog LU decomposition where possible, while one wants to avoid solving large number of ODEs and / or a couple PDEs by digital methods.

V. Starting towards a Unified Analog Computing Framework

So far most of the computation described has been challenges of primarily feedforward computation. In such cases, comparison based on fundamental operations (e.g. MAC), even in neuromorphic systems seems appropriate. Figure 5 shows analog computation includes a wider space including optimization over potential or energy surfaces, recurrent networks, and solutions to spatio-temporal (e.g. PDE) problems. Hopfields foundational work involved the dynamics of these recurrent networks modeling these energy solutions [1], [2], followed on by related network analysis efforts [74], [73]. The energy function is used as a medium to implement an optimization problem on a feedback network [74]. These efforts were applied to multiple optimization problems, including solving the TSP [6]; solving this problem opens the possibility of solving this system in polynomial time using physical computing. These approaches relate to the ARNN super-Turing computability model of Seigelman [87], [26]. Recent discussions around ODE solutions to the 3-SAT problem [75], [76], as well as the FPAA implementations for L_1 norm minimization [77] further open these questions. Further, recent results between analog and neuromorphic engineering communities demonstrated optimal path planning in an polynomial size array of neurons in polynomial time [71]. These results are beyond the typical improvements in processor component efficiencies mentioned in Section III. It seems we are at a time to reopen the question of whether NP problems could be solved in polynomial time, but in this case, not by a digital (Turing) machine, but using a physical computing machine.

Quantum computing, in Shor and Grovers algorithms, has theoretically shown computational capability beyond Turing limits. Multiple hypotheses show that the form of quantum computing used could equally well be done through analog computing [79], [80], [81], particularly considering Hilbert space computing with analog circuits [82]. Recently, a discrete, analog, bench-top implementation for a small quantum system was demonstrated [83], [84], [85]. The fundamental computation is a modified Fourier transform, where understanding the algorithm might enable alternate, efficient analog implementations as is already done for DCT or DFT computations [78]. But fundamentally we see another physical computing space pushing against generally accepted digital Turing machine limits, potentially enabling new algorithmic opportunities.

Although we see only dimly at this point, what we can see points towards significantly greater computation capabilities due to physical approaches. These devices acting on Real-valued quantities, both as I/O and internal stored quantities (e.g. state variables). No doubt we have noise, the number of particles are finite, etc, but none of these issues fundamentally takes away the potential of real-valued numbers and operations within joint space and time. Figure 6 shows the contrast between digital and analog computation models. As a result, one might be able to start pondering the possibility of a real-valued or analog Turing machine model, and what potential computational and algorithmic opportunities are possible.

VI. Conclusion

As we have seen throughout this paper, it is time to renew these early beverage discussions, now bringing the results into serious discussions. Significant programmable and configurable physical (analog) computing systems do exist. The computational framework for analog computation is growing; the widespread emergence of analog computing hardware will only accelerate this process. The current state of analog computation includes energy efficient computation, and analog numerical analysis, moving towards starting a unified analog-computing framework as part of physical computing. Not solving these issues will hinder the progress of analog computation, and computation in general. The computability of analog computation, potentially described through an analog Turing machine, opens up new questions of the computability of analog systems, including asking whether these results could eventually show NP class problems could be solved in polynomial (P) time by physical computing.

References

[1] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of NationalAcademy of Science*, vol. 79, 1982, pp. 2554.

[2] J.J. Hopfield, "Neurons with graded responses have collective computational properties like those of two- state neurons," *Proceedings of NationalAcademy of Science*, vol. 81, 1984, pp. 3088-3092.

[3] C. Mead, "Analog VLSI and Neural Systems," *Addison Wesley*, 1989.

[4] G. E. Moore, "Cramming more components onto integrated circuits," Electronics, Vol. 38, no. 8, 1965, pp.

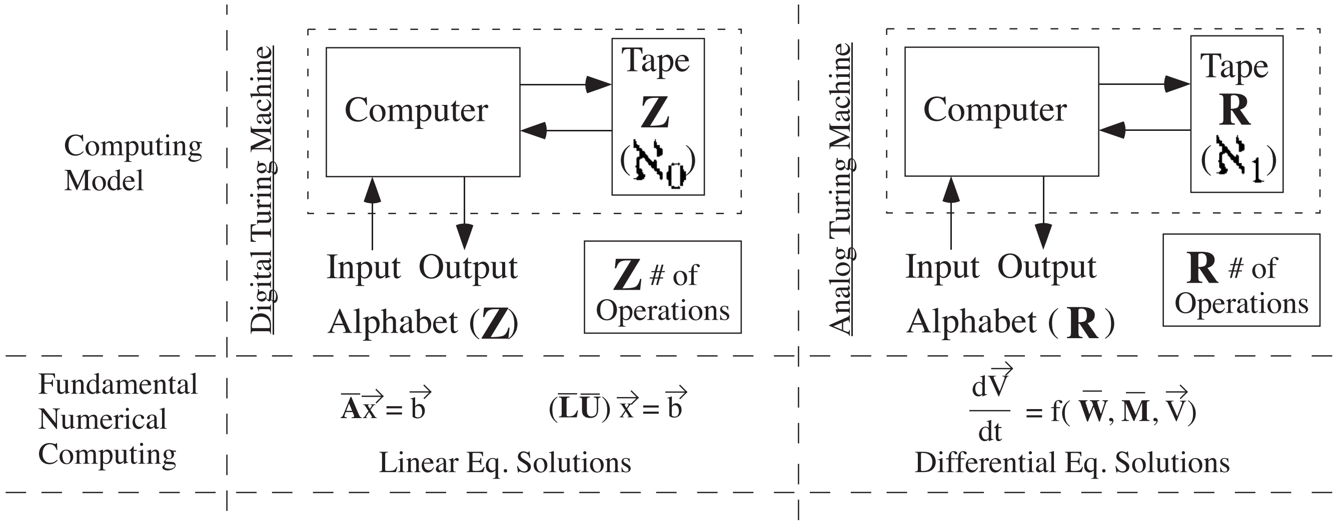
[5] G. E. Moore, Progress in Digital Integrated Electronics, IEEE IEDM, 1975, pp. 11-13.

[6] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, *Biological Cybernetics*, vol. 52, 1985, pp. 141-152.

[7] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R, Wunderlich, S. Nease, and S. Ramakrishnan, "A Programmable and Configurable Mixed-Mode FPAA SoC," *IEEE Transactions on VLSI*, vol. 24, no. 6, 2016, pp. 2253-2261.

[8] R. Turing, "On Computable Numbers," *Proceedings of the London Mathematical Society*, 1937, pp. 230265.

[9] "Excerpts from A Conversation with Gordon Moore: Moores Law," Video transcript, Intel, 2005.



$$\bar{\mathbf{A}}\vec{x} = \vec{b} \qquad (\bar{\mathbf{L}}\bar{\mathbf{U}})\,\vec{x} = \vec{b}$$

$$\frac{d\vec{V}}{dt} = f(\bar{\mathbf{W}}, \bar{\mathbf{M}}, \vec{V})$$

Fig. 6. Digital and Analog Computing models and resulting numerical computing approaches. Digital numerical computation moves problems towards solutions of linear equations; analog numerical computation moves problems towards solutions of differential equations. One could imagine Turing Machines for digital and analog solutions, where digital operates over countable (integer) sets, and analog operates over sets similar to the real number line.

[10] B. Hoeneisen and C. A. Mead, "Fundamental Limitations in Microelectronics – I. MOS Technology," Solid State Electronics, vol. 15, no. 7, 1972, pp. 819-829.

[11] B. Hoeneisen and C. A. Mead, "Current-voltage characteristics of small size MOS transistors," *IEEE Transactions on Electron Devices*, vol. 19, no. 3, 1972, pp. 382-383.

[12] Carver Mead and Lynn Conway "Introduction to VLSI System Design,", Addison-Wesley, 1980. ISBN 0-201-04358-0. Early drafts found at http://ai.eecs.umich.edu/people/conway/VLSI/VLSIText/VLSIText.html.

[13] W. Thomson, "Mechanical integration of the general linear differential equation of any order with variable coeffcients," *Proceedings Royal Society of London*, vol. 24, 1876, pp. 271275.

[14] D. M. MacKay and M. E. Fisher, *Analogue Computing at Ultra-High Speed: An Experimental and Theoretical Study*, Wiley and Sons, NY, 1962.

[15] W. J. Karplus, *Analog Simulation: Solution of Field Problems*, McGraw Hill, NY, 1958.

[16] C. Shannon, "Mathematical theory of the differential analyzer," *Journal Mathematical Physics, MIT*, vol. 20, 1941, pp. 337354.

[17] M. Pour-El, "Abstract computability and its relation to the general purpose analog computer," *Transactions on American Mathematical Society*, vol. 199, 1978, pp. 128.

[18] L. Lipshitz, and L.A. Rubel, "A differentially algebraic replacement theorem, and analog computability," *Proceedings American Mathematical Society*, vol. 99, 1981, pp. 367372.

[19] B. J. MacLennan, "A Review of Analog Computing, Technical Report UT-CS-07-601, Dept. of ECE, U. of Tennessee, Knoxville, Sept. 13, 2007. www.cs.utk.edu/ mclennan.

[20] C. Schlottmann and J. Hasler, "High-level modeling of analog computational elements for signal processing applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 19451953, 2014.

[21] M. Collins, J . Hasler, and S. George, "An Open-Source Toolset Enabling Analog–Digital Software Codesign," *Journal of Low Power Electronics Applications*, vol. 6, no. 1, January 2016.

[22] C. Mead, "Neuromorphic electronic systems," *Proceedings of IEEE*, vol. 78, 1990, pp. 1629-1636.

[23] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128 x 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," *IEEE Custom Integrated Circuits Conference*, October 2004, pp. 651-654.

[24] H.T. Siegelmann, "Computation Beyond the Turing Limit," *Science*, vol. 238, no. 28, 1995, pp. 545-548.

[25] H. T. Siegelmann and Shmuel Fishman, "Analog computation with dynamical systems," *Physica D*, vol. 120, 1998, pp. 214-235.

[26] J. Cabessa and H. T. Siegelmann, "The Super-Turing Computational Power of Plastic Recurrent Neural Networks," *International Journal of Neural Systems*, vol. 24, no. 8, 2014, pp. 1450029-1 – 1450029-22.

[27] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, "Single transistor learning synapses," in Gerald Tesauro, David S. Touretzky, and Todd K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge, MA, 1994, pp. 817-824.

[28] P. Hasler, C. Diorio, B. Minch, and C. Mead, "Single transistor learning synapse with long term storage," *IEEE International Symposium on Circuits and Systems*, vol. 3, May 1995, pp. 16601663.

[29] P. Hasler, B. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate devices," *in Advanced Research in VLSI*, 1999, pp. 215229.

[30] S. Kim, J. Hasler, and S. George, "Integrated Floating-Gate Programming Environment for System-Level ICs," *IEEE Transactions on VLSI*, vol. 24, no. 6, 2016, pp. 2244-2252.

[31] C. Schlottmann, and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal of Emerging CAS*, vol. 1, 2012, 403-411.

[32] C. Twigg, J. Gray, and P. Hasler, "Programmable Floating-gate FPAA switches are not dead weight," *International Symposium on Circuits and Systems*, May 2007, pp. 169-72.

[33] J. Hasler and S. Shah, "Learning for VMM + WTA Embedded Classifiers," GOMAC, March 2016.

[34] J. Hasler and S. Shah, "Learning for Analog VMM + WTA Embedded Classifiers Demonstrated on a Configurable Platform," submitted to *IEEE Transactions on VLSI*, February 2016.

[35] S. Ramakrishnan and J. Hasler, "Vector-Matrix Multiply and WTA as an Analog Classifier," IEEE TVLSI, vol. 22, no. 2, 2014, pp. 353-361.

[36] J. Lazzaro, S. Ryckebusch, M. A. Mahowald and C. A. Mead, Winner-take-all networks of O(N) complexity, in Advances in Neural Information Processing Systems 1, Morgan Kaufmann Publishers, CA, 1989.

[37] Marvin Minsky and Seymour Papert, *Perceptrons: An Introduction to Computational Geometry*, The MIT Press, Cambridge MA, 1969.

[38] V. Srinivasan, D. Graham, and P. Hasler, "Floating-gates transistors for precision analog circuit design: an overview," *Midwest Symposium on Circuits and Systems*, vol.1., Aug. 2005, pp. 7174.

[39] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 2, pp. 280-291, Feb. 2007.

[40] V. Srinivasan, G. Serrano, C. Twigg, and P. Hasler, A Floating-Gate-Based Programmable CMOS Reference, *IEEE Transactions on Circuits and Systems I*, Vol. 55, No. 11, pp. 3448 - 3456, Dec. 2008.

[41] S.Y. Peng, M. S. Qureshi, P. Hasler, A. Basu, andF. L. Degertekin, "A Charge-Based Low-Power High-SNR Capacitive Sensing Interface Circuit," *IEEE Transactions on Circuits and Systems I*, vol. 55, No. 7, August 2008,pp. 1863

[42] D.W. Graham, P. Hasler, R. Chawla, and P.D. Smith, "A low-power, programmable bandpass filter section for higher-order filter applications,"

IEEE Transactions on Circuits and Systems I, Vol. 54, no. 6, pp. 1165 - 1176, June 2007.

[43] R. Chawla, F. Adil, G. Serrano, and P. Hasler, Programmable Gm-C filters using floating-gate operational transconductance amplifiers, IEEE Transactions on Circuits and Systems I, vol. 54, no. 3, pp. 481-491, Mar 2007.

[44] G. Serrano, M. Kucic, and P. Hasler, “Investigating programmable floating-gate digital-to-analog converter as single element or element arrays,” *Midwest Symposium on Circuits and Systems*, vol. 1, August 2002, pp. 7577.

[45] P.Brady and P.Hasler, “Offset compensation in flash ADCs using floating-gate circuits,” *IEEE International Symposium on Circuits and Systems*, vol. 6, May 2005, pp. 61546157.

[46] A. W. Pereira, D. J. Allen, and P. E. Hasler, “A 0.5μm CMOS programmable discrete-time Delta-Sigma modulator with floating gate elements,” *International Symposium on Circuits and Systems*, vol. 1, May 2004, pp. 213216.

[47] P. Hasler, “Low-power programmable signal processing”, *Int. Workshop on System-on-Chip for Real-Time Applications*, 2005, pp. 413 – 418.

[48] M. Kucic, A. Low, P. Hasler, and J. Neff, “A programmable continuous-time floating-gate Fourier processor,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, 2001, pp. 90-99.

[49] S.-Y. Peng, P. Hasler, and D. Anderson, “An analog programmable multi-dimensional radial basis function based classifer,” *VLSI - SoC IFIP International Conference on Very Large Scale Integration*, 2007, pp. 13 - 18.

[50] S.-Y. Peng, B. A. Minch, and P. Hasler “Analog VLSI Implementation of Support Vector Machine Learning and Classication,” *ISCAS* 2008.

[51] P. Hasler and J. Dugger, “An analog floating-gate node for supervised learning,” *IEEE Transactions on Circuits and Systems I* , vol. 52, no. 5, pp. 834845, 2005.

[52] P. Hasler, P. D. Smith, D. Graham, R. Ellis, and D. V. Anderson, “Analog Floating–Gate, On–Chip Auditory Sensing System Interfaces”, *IEEE Transactions on Sensors*, 2005.

[53] A. Bandyopadhyay, P. Hasler, and D. V. Anderson, “A CMOS Floating-Gate Matrix Transform Imager,” *IEEE Transactions on Sensors*, 2005.

[54] A. Bandyopadhyay, J. Lee, R. Robucci, and P. Hasler, “MATIA: a programmable 80 W/frame CMOS block matrix transform imager architecture,” *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 663672, March 2006.

[55] R. Wunderlich, F. Adil, and P. Hasler, “A Floating-Gate-Based Field-Programmable Array of Analog and Digital Devices,” *IEEE Transactions on VLSI*, 2013.

[56] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, “A Digitally-Enhanced Reconfigurable Analog Platform for Low-Power Signal Processing,” *IEEE Journal of Solid State Circuits*, October 2011.

[57] S. Koziol and P. Hasler, “Reconfigurable Analog VLSI Circuits for Robot Path Planning,” *NASA/ESA Conference on Adaptive Hardware and Systems*, 2011

[58] S. Koziol, D. Lenz, S. Hilsenbeck, S. Chopra, P. Hasler, and A. Howard, “Using Floating-Gate Based Programmable Analog Arrays for Real-Time Control of a Game-Playing Robot,” *IEEE Systems Man and Cybernetics conference*, Oct. 2011.

[59] G. Frantz, and R. Wiggins, “ Design case history: speak and spell learns to talk,” *IEEE Spectrum*, vol 19, 1982, pp. 4549.

[60] B. Marr , B. Degnan, P. Hasler, and D. Anderson, “Scaling Energy Per Operation via an Asynchronous Pipeline,” *IEEE Transactions on VLSI*, Issue 99, 2012, pp. 1-5.

[61] J. Hasler and H. B. Marr, Finding a Roadmap to achieve Large Neuromorphic Hardware Systems, Frontiers in Neuromorphic Engineering, September 2013. pp. 1-29.

[62] J. Hasler and H. B. Marr, Implementation of Large Neuromorphic Hardware Systems, GOMAC, 2015.

[63] J. Hasler, Energy Constraints for Building Large-Scale Neuromorphic Systems, GOMAC, March 2016.

[64] E. Farquhar and P. Hasler, “A bio-physically inspired silicon neuron, *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 3, pp. 477-488, March 2005.

[65] S. Ramakrishnan, P. Hasler, and C. Gordon, Floating gate synapses with spike-time-dependent plasticity, IEEE Transactions on Biomedical Circuits and Systems, vol. 5, no. 3, pp. 244252, 2011.

[66] S. Nease, S. George, P. Hasler, S. Koziol, and S. Brink, “Modeling and implementation of voltage-mode CMOS dendrites on a reconfigurable analog platform,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 1, pp. 76-84, 2012.

[67] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, “A learning- enabled neuron array IC based upon transistor channel models of biological phenomena,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71-81, 2013.

[68] S. Ramakrishnan, R. Wunderlich, J. Hasler, and S. George, Neuron array with plastic synapses and programmable dendrites, IEEE Transactions on Biomedical Circuits and Systems, vol. 7, no. 5, pp. 631-642, 2013.

[69] S. George, J. Hasler, S. Koziol, S. Nease, and S. Ramakrishnan, “Low power dendritic computation for wordspotting,” Journal of Low Power Electronics Applications, vol. 3, 2013, 73-98.

[70] S. Koziol, P. Hasler, and M. Stilman, Robot path planning using field programmable analog arrays, IEEE Conference on Robotics and Automation (ICRA), 2012, pp. 1747-1752.

[71] S. Koziol, S. Brink, and J. Hasler, “A neuromorphic approach to path planning using a reconfigurable neuron array IC,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2724-2737, 2014.

[72] J.J. Dongarra, P.Luszczek, and A. Petitet, “The LINPACK Benchmark: past, present and future,” *Concurrency and Computation: Practice and Experience*, Wiley, 2003, pp. 803820.

[73] A.E Atiya, Y.S. Abu-Mostafa, “An analog feedback associative memory,” *IEEE Trans. Neural Networks*, vol. 4, 1993, pp. 117-126.

[74] Y. S. Abu-Mostafa, “Information theory, complexity, and neural networks,” *IEEE Communications Magazine*, 1989, pp. 25-28.

[75] S. Cocco and R. Monasson, “Analysis of the computational complexity of solving random satisfiability problems using branch and bound search algorithms, 2013.

[76] R. Sumi, M. Varga, Z. Toroczkai, and M. Ercsey-Ravasz, “Order-to-chaos transition in the hardness of random Boolean satisfiability problems, 2016.

[77] S. Shapero, A. Charles, C. Rozell, and P. Hasler, Low power sparse approximation on reconfigurable analog hardware, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 2, no. 3, pp. 530-541, 2012.

[78] S. Suh, A. Basu, C. Schlottmann, P. Hasler ,and J .Barry, “Low-power discrete Fourier transform for OFDM: A programmable analog approach,” *IEEE Transactions on Circuits and Systems I*, vol. 58, no. 2, pp. 290298, 2011.

[79] D.K. Ferry, R. Akis, and J. Harris, “Quantum wave processing,” *Superlattices and Microstructures* vol. 30, no. 81, 2001.

[80] D.K. Ferry, R. Akis, I. Knezevic, “Quantum wavesthe proper basis for low dissipation quantum computing,” *Microelectronic Engineering*, vol. 63, 2002, pp. 1721.

[81] D. K. Ferry, R. Akis, M. J. Gilbert and I. Knezevic, ”Do we need quantum for quantum computing?”, *SPIE 5115 - Noise and Information in Nanoelectronics, Sensors, and Standards*, May 9, 2003.

[82] L. B. Kish, “Hilbert Space Computing by Analog Circuits,” *Proceedings of SPIE 5115 - Noise and Information in Nanoelectronics, Sensors, and Standards*, May 9, 2003.

[83] B. R. La Cour and E. C. G. Sudarshan, “Classical model for measurements of an entanglement witness,” *Physical Review A*, vol. 92, 2015. pp. 032302-1 – 032302-7.

[84] B. R. La Cour and G. E. Ott. “Signal-based classical emulation of a universal quantum computer,” *New Journal of Physics*, May 13, 2015. pp. 1-19. DOI: 10.1088/1367-2630/17/5/053017

[85] B. R. La Cour, C. I. Ostrove, G. E. Ott, M. J. Starkey and G. R. Wilson, “Classical emulation of a quantum computer,” *Inter. Journal of Quantum Information*, Vol. 14, No. 1, April 2016, pp. 1640004-1–12.

[86] A. Ben-Hur, A. Roitershtein, and H. T. Siegelmann, “On Probabilistic Analog Automata,” *Journal Theoretical Computer Science*, Vol. 320, no. 2-3, 2004. pp. 449 - 464.

[87] H. T. Siegelmann, “Turing on Super-Turing and Adaptivity,” Prog. Biophys Mol. Biol, vol. 113, no. 1, 2013, pp. 117-126.

[88] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*, Wiley, New York, 2nd edition, 2001.

[89] C. Farabet, P. Martini, Akselrod, S. Talay, Y. LeCun, and E. Culurciello, “Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems,” *IEEE International Conference on Circuits and Systems (ISCAS)*, 2010.