Journal of
Low Power Electronics and Applications

MDPI

Article

Physical Computing: Unifying Real Number Computation to Enable Energy Efficient Computing

Jennifer Hasler * and Eric Black

Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332-250, USA; 3ric.black@gmail.com
* Correspondence: jennifer.hasler@ece.gatech.edu; Tel.: +1-404-894-2944; Fax: +1-404-894-4641

Abstract: Physical computing unifies real value computing including analog, neuromorphic, optical, and quantum computing. Many real-valued techniques show improvements in energy efficiency, enable smaller area per computation, and potentially improve algorithm scaling. These physical computing techniques suffer from not having a strong computational theory to guide application development in contrast to digital computation's deep theoretical grounding in application development. We consider the possibility of a real-valued Turing machine model, the potential computational and algorithmic opportunities of these techniques, the implications for implementation applications, and the computational complexity space arising from this model. These techniques have shown promise in increasing energy efficiency, enabling smaller area per computation, and potentially improving algorithm scaling.

Keywords: physical computing; analog computing; complexity theory

Citation: Hasler, J.; Black, E. Physical Computing: Unifying Real Number Computation Enabling Energy Efficient Computing. *J. Low Power Electron. Appl.* **2021**, *11*, 14. https://doi.org/10.3390/jlpea11020014

Academic Editor: Andrea Acquaviva

Received: 18 February 2021
Accepted: 22 March 2021
Published: 26 March 2021

1. Introducing Physical Computing and Physical Turing Machine Modeling

The rapid progress in today's ubiquitous programmable digital infrastructure relies as much on digital Turing machine theory [1] as it has on Moore's law scaling [2–4] and VLSI [5] in providing a framework to use these hardware capabilities. A roadmap of future directions was in place as new technologies were available. Turing Machine theory (Figure 1) provides a core digital computation, computes over discrete, integer values, and was based upon bookkeeping businesses at the time [6]. The mathematical framework is central to abstracting digital computations, central to computer architectures and algorithms, as well as is central to numerical computation and analysis (Figure 1).

Not all computation operates over discrete, integer values, computational approaches that have used various physical phenomena for potentially lower power/energy and smaller area applications. Mead's original hypothesis (1990) [7] predicted analog processor area will decrease at least 100× in size compared to digital computation, and energy consumed will decrease at least 1000× compared to digital computation. These techniques were experimentally demonstrated through Vector-Matrix Multiplication (VMM) in custom (2004) [8] and in configurable large-scale Field Programmable Analog Arrays (FPAA) (2012) [9] as well as in many later systems and configurations. Following the analog computing efficiencies over digital techniques, neuromorphic computing formulated and achieved significant efficiencies over digital computing (2013) [10]. Recently, quantum computing has advocates claiming its supremacy over digital computing, stating there exists experimental problems where quantum computing is demonstrably more capable than digital computing (2019) [11]. Some debate these claims (e.g., [12]). This computationally efficient quantum-computing noise generator has direct parallels to an analog noise generator and computing [13], illustrating a relationship between these real-valued computations. Any real-valued computations will show computational efficiencies over digital ones. We define Physical Computing as computing using real-valued quantities, quantities that can be amplitude, space, or time, Physical computing includes Analog, Quantum, Neuromorphic,

and Optical techniques. This work both defines and further identifies the relationships between forms of Physical Computation.

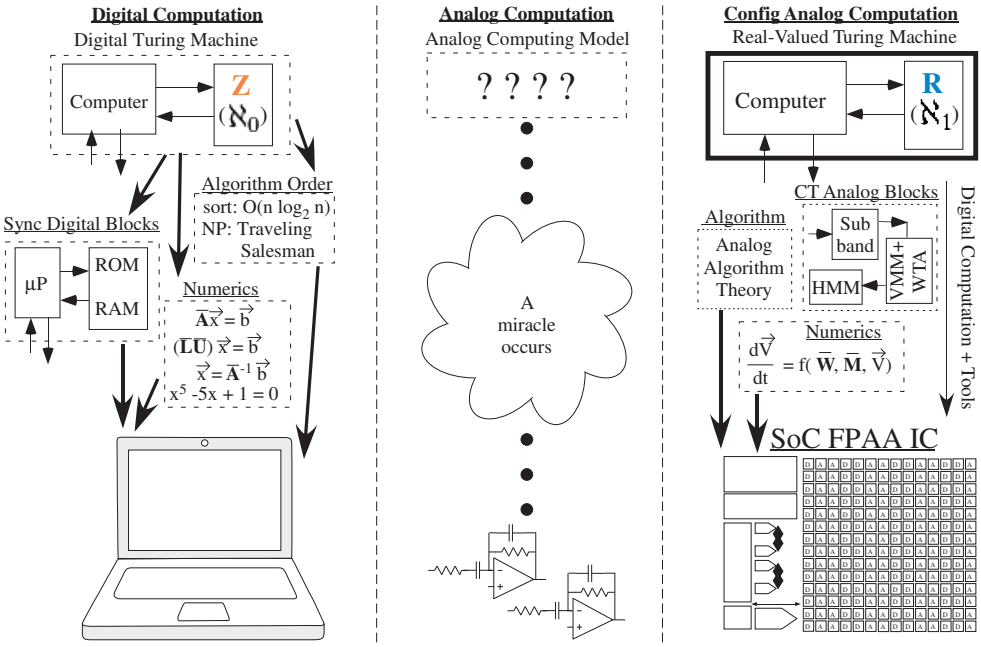

Figure 1. Digital Computation builds from the framework of Turing Machines, setting up capability of computer architectures, computer algorithms, and resulting numerical analysis, being the basis for our day to day digital computing. Classical Analog Computation is perceived to have little computational modeling, as well as architectures and algorithms, seeming to be bottom-up artwork rather than top-down digital computing design. Configurable Analog Computation, a Physical Computing real-valued computing technique originally FPAA enabled, builds on recent framework in architectures, algorithms, abstraction, and numerical analysis. This approach enables a Physical Turing Machine model unifying real-value computation.

Energy efficiency, as well as other computational metrics, are often described using order notation such as $O(\cdot)$. Typically physical computing techniques typically focus on improving the coefficient for the $O(\cdot)$ metric. As mentioned above, the coefficient for computational energy efficiency, the energy or power required per operation, decreases by typically $1000\times$ an equivalent digital computation, while the scaling metric in $O(\cdot)$ remains unchanged. Some aspects might improve the polynomial or similar function inside of $O(\cdot)$ say by architectural improvements in the physical computing structure (e.g., [14]). This effort affirms these improvements in energy efficiency, and builds upon these opportunities in energy efficiency by discussing the possibility to significantly change the scaling dependency characterized by $O(\cdot)$ for physical computing systems because of the different computing capabilities enabled by providing a model and framework to explore these opportunities.

For Analog computing, more carefully known as analog electronic computing and the most experimentally developed of Physical techniques, has lacked a theoretical framework and high-level computing model. Classical Analog computation is widely perceived as a bottom-up design approach practiced by a few artistic masters (Figure 1). The goal was to formulate a problem in a few Ordinary Differential Equations (ODE), and have a circuit master design the system. Once this miracle occurs, the one particular system becomes functional [15]. These design techniques do not scale to a wide user capability as they do for digital computation. There are no classic textbooks explaining analog system synthesis. The classical situation is less optimistic for neuromorphic, quantum, or optical computing.

Recently, analog computation has developed a framework (Figure 1) that includes analog numerical analysis techniques [16], analog algorithm complexity theory [14], and

analog algorithm abstraction theory [17]. Current programmable Analog design is no longer seen as numerically inferior to digital computation [16], or seen as too complex to have levels of abstraction like digital computation [17], or seen to be governed by digital processor techniques where it rather now pushes the questions for both analog and digital architecture questions [14]. Analog computation becomes relevant with the advent of programmable and configurable FPAA devices [18,19] and the associated design and synthesis tools [20,21] incorporating parts of this framework.

The recently developing analog frameworks achieve a level where one can ask realistic questions, as well as realistically propose, an analog computing model. This work proposes a Physical Turing machine as a machine operating over real-valued quantities as well as operating over real-valued sequences over real-valued timesteps. Physical computing utilizes one or more representations and internal variables encoded as real values, including amplitude, voltage, current, time, or space. This work connects all Physical computing techniques (e.g., analog, neuromorphic, optical, and quantum) through a single computing framework, enabling one, or a mixture, of these techniques as well as provides a bridge to use the results from one technique towards another technique. This Physical Computing model builds from experimental techniques and measurements as well as frameworks for their effective implementation, rather than hypothesizing unverifiable theoretical concepts.

This paper addresses Physical Computing and its associated Turing machine model addressing the capability of real-valued computation, as in physical computation, as compared with integer-valued computation, as in digital computation. The discussion starts by addressing the algorithmic complexity of real-valued and integer valued computation (Section 2). The discussion then establishes that the physical computing substrate is real-valued even in the presence of noise (Section 3), a necessary condition for real-valued physical computing. The discussion shows the connection and translation between quantum qubit computation and an analog system modeling that computation (Section 4). The discussion generalizes physical computing, and the equivalence between approaches (Section 5), finally discussing the the implications and opportunities of Physical computing (Section 6) and summarizing this discussion (Section 7).

2. Algorithmic Complexity of Real-Valued Computation

Physical computing techniques compute over real-valued variables (**R**), representations, and timescales. Analog computing can compute over continuous-valued voltage or current state variables (e.g., 1 V to 3 V) as well as computing over continuous-valued timesteps. Neuromorphic computing, which includes neuromorphic biological or electronic computing, involves similar capabilities to analog computing, at least over limited regions, as well as continuous-spatial regions at least over limited regions (e.g., dendritic cables). Optical computing also involves continuous amplitudes operating over continuous-valued timesteps with two-dimensional (2D) and three-dimensional (3D) continuous spatial dimensions. Quantum computing operates using a continuous probabilities modeling continuous-time and continuous 3D space, as well as continuous superposition mixtures of two or more states. All of these techniques compute effectively utilizing at least one real-valued dimension.

Real valued functions in multiple dimensions have the same complexity as a real-valued function in one dimension. The size of the set of all real numbers ((**R**) between 0 and 1 is infinitely larger than all integers (**Z**) between 0 and ∞. The size of **Z** between 0 and ∞ is represented as $\aleph_0$, and the size bf R between 0 and 1 is represented at $\aleph_1$. Two **R** dimensions can fit into a single **R** dimension. Working along the diagonals of a 2D **R** map, one can recount the 2D space into a single one-dimensionsal (1D) space (Figure 2) as the same order of infinity ($\aleph_1$). More available dimensions opens up additional implementation opportunities and resulting efficiencies, although these dimensions do not affect the system complexity.

A real-valued Physical Turing Machine models computing over **R** values and timesteps, similar to a Digital Turing Machine models computing over **Z** values and timesteps (Figures 1 and 3).

A Digital Turing machine computes over a **Z** set of input and output alphabets with **Z** internal variables and **Z** size tape operating over **Z** timesteps. The theoretical Turing Machine model does not require the computation follows the same approach, and yet, digital computing often resembles parts of the Turing Machine techniques. To fully model the continuous-time (CT) computation over **R** values, a Physical Turing Machine Model computes over a **R** set of input and output alphabets with **R** internal variables and **R** size tape (internal memory) operating over **R** timesteps. Utilizing **Z** size input and output alphabets for a Physical Computation is still modeled by the Physical Turing Machine. Physical computing operates over $\aleph_1$ (∞ for **R**) as opposed to synchronous digital computing operating over $\aleph_0$ (∞ for **Z**). Model allows directly extending known properties and theorems for **Z**-valued Turing machines to these **R**-valued Turing machines. Although one could consider a physical implementation specific model (e.g., [22,23]), it likely misses the wider computational space and becomes harder to generalize across all **R**-valued computing, as well as requiring to build an entirely new theoretical infrastructure.

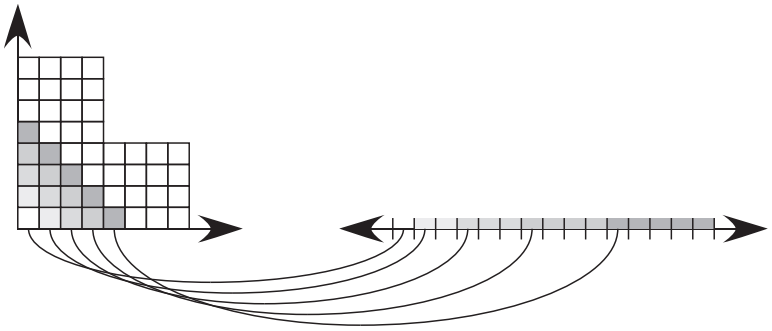

Figure 2. One can map a two dimensional infinity, whether countable or real, to a one-dimensional infinity of the same order (countable or real, respectively) by projecting the successive diagonals of the two-dimensional space into the one-dimensional space. Each box is as small a region possible (real or countable), and they project into small regions into the one-dimensional space. The difference between real or countable is the size of these regions.

Deutsch began to develop an understanding of physical computing models as part of their wrestling to understand the nature of quantum computing [24]. These discussions strengthen and generalize Deutsch's wrestling with opportunities in quantum computing. Deutsch attempts to generalize Turing's classical definition [1] towards physical approaches

> "Every finitely realizible physical system can be perfectly simulated by a universal model computing machine operating by finite means" (p. 99) [24].

where he states a **Z**-valued Turing machine cannot perfectly simulate a classical dynamical system, as well as he recognizes the impact of non-decreasing entropy (e.g., loss) impacts these computations. Deutsch only imagines **Z**-valued input and output alphabets, while he opens the possibility of computing over a continuum of values, which is more concretely defined as computation over **R**-values, and then moves that a quantum dynamics provides a means towards reaching these opportunities, while mostly missing this opportunity in **R**-valued systems. He mathematically attempts to show non-dynamical quantum operations could be operating over a continuum, inspired by the introduction of quantum computing by Feynman [25,26]. A model that computes with **R**-valued alphabets also simplifies to computing with **Z**-valued input and output alphabets. The approach in this discussion generalizes **R**-valued computation for the continuum of classical and quantum physics, heavily based in decades of physical (e.g., analog, neuromorphic) computing.

Deutsch moves to extend the Church-Turing principle to be related to a quantum physical system. This discussion also moves to effectively extend the Church-Turing principle,

> "Every 'function which would naturally be regarded as computable' can be computed by the universal (**Z**-valued) Turing machine"—Turing [1]

with the explicit restatement of the Church-Turing thesis for-**R**-valued computation:

> "Every finitely realizible 'function which would naturally be regarded as computable' can be computed by the universal (**R**-valued) Turing machine"

where the **Z**-valued Turing machine ⊂ **R**-valued Turing machine. Finite means, which includes finite resources as well as finite amount of time, is essential to any practical physical computation.

Algorithmic complexity between Synchronous Digital Computation and Physical computation is the comparison between **R** versus **Z** Turing capabilities (Figure 3). Algorithm complexity (Figure 3) considers whether a particular computing structure can compute certain algorithms in Polynomial (P) time, or scales by some other function, such as exponential time (EXPTIME). One can consider the class of polynomial-time physical (P_A) and digital (P_D) algorithms, the class of digital NP problems (NP_D) and similar class of NP problems for analog computation (NP_A), as well as the class of exponential-time analog ($EXPTIME_A$) and digital ($EXPTIME_D$) algorithms (Figure 3). One must consider a product of time and resources, although if one considers only polynomial resources, time complexity is sufficient. P_A completely overlaps with P_D as one can make digital gates from analog blocks (Figure 3).

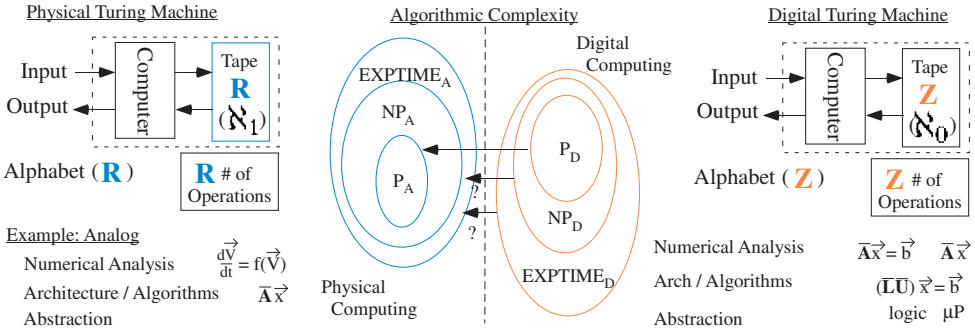

Figure 3. Physical (Real-Valued, **R**) and Digital (Integer-Valued, **Z**) computation comparison. Physical computing includes quantum, optical, analog and neuromorphic approaches. Polynomial time (P) digital algorithms are part of P physical algorithm space. Non-polynomial time (NP, EXPTIME) digital algorithms might be part of P physical algorithm space.

The open question is how does NP_D and $EXPTIME_D$ overlap with P_A, NP_A and $EXPTIME_A$ (Figure 3)? Does P_A have any overlap with NP_D or even $EXPTIME_D$? If part of $NP_D \subset P_A$, then a Physical computing system would solve at least one NP_D application in P. Does P_A extend into uncomputable spaces by digital Turing machines (e.g., halting problem)? This framework will unify previous physical computing techniques and algorithms. Hopfield's work solving the Traveling Salesman Problem (TSP) [27] opens one's imagination that eventually NP_D could be in solved in P_A through recurrent networks modeling optimization problems minimizing energy surfaces [27–31], Hava Siegelmann created a number of useful theoretical discussions for analog computing, arguing analog P_A has computational capabilities beyond P_D by minimizing energy surfaces (ARNN model) [23,32]. Multiple coupled ODEs systems have been theoretically proposed to solve NP_D (e.g., 3-SAT [33,34]) in P_A that could be implemented on a continuous-time analog platform. Quantum computing (e.g., Shor and Grover's algorithms) has theoretically shown P_A is larger than P_D [35,36]. Often the Church-Turing conjecture is interpreted to mean that physical computing techniques (Figure 3) are equivalent to a single machine processing a memory tape, which requires countable (**Z**) input and output alphabets. This interpretation is far too restrictive of physical computing approaches.

A unified analog computing framework (Figure 3) equates transformations between techniques, allowing a solution in one space could be translated to another (Figure 4). For example, a quantum computation could be transformed to room temperature analog computation. The transformations allow all technologies to be utilized in spaces and applications where they naturally have physics advantages.

Transformations between physical computing and digital computing illustrate the differences between these computing mediums, showing significant differences between

R verses **Z** computation. Transformations from **Z** to **R** are straight-forward, and one expects a physical system to implement computations over integer (e.g., digital) values. Transformations from **R** to **Z** computation require interpolations to, or numeric approximations from **R** to **Z** (Figure 4). Synchronous digital simulation results of physical computation must always be approached cautiously, as digital computation is more limited than the resulting physical substrate (Figure 4). Physical ODEs are solved in $\aleph_1$, with **R**-valued timesteps compared to **Z**-valued timesteps by digital emulation. Multiple-timescale nonlinear ODEs (Figure 4) inherently utilize **R**-valued timesteps to directly handle exponentially fast moving nonlinear physical system dynamics. Analog WTA physical system uses two (or more) competitively strong nonlinearities, resulting in consistent analog solutions while creating a difficult ODEs to solve numerically (e.g., [37]). The error of digital ODE solutions are limited as $O(f^{k+1}(\cdot))$ for a kth order solver (RK45 is 4th order) [38], so high derivatives, like exponential functions destroy the accuracy and convergence of digital ODE solutions. Attempting to validate physical computing systems (e.g., systems of ODEs) through digital numerical approximations creates the unnecessary concern over having exponentially fast moving nonlinear results for a physical system that are very real for synchronous digital computation. Physical algorithms must be developed and verified only through physical hardware, and discrete simulation or analysis of physical algorithms cannot invalidate potential results.

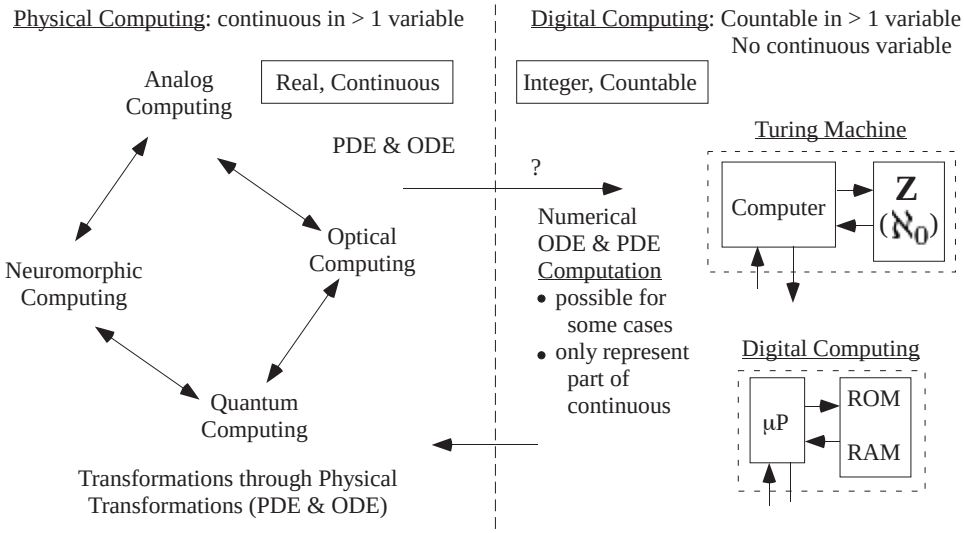

Figure 4. Equivalences between physical computing techniques (real variables) and the challenges equating these approaches digital (integer variables). An integer-valued computational framework can approximate a real valued computation, and may work within reasonable bounds for a range of cases, but will not be successful within a reasonable amount of time (e.g., exponential algorithm scaling) in other cases.

Competing exponentially increasing and decreasing functions converge effectively using physical computing to solve the ODE functions, tasks that are extremely difficult for **Z**-based computing. For example, a Winner-take-all (WTA) circuit (Figure 5) operating with subthreshold currents utilizes competing exponential functions modeled by the transistor current-voltage relationships:

$$\begin{cases} C\frac{dV_1}{dt} = I_1 - I_{s01}e^{\kappa\Delta V/U_T}e^{\sigma\Delta V_1/U_T} \\ C\frac{dV_2}{dt} = I_2 - I_{s01}e^{\kappa\Delta V/U_T}e^{\sigma\Delta V_2/U_T} \\ C\frac{dV}{dt} = I_{s0}\left(e^{\kappa\Delta V_1/U_T} + e^{\kappa\Delta V_2/U_T}\right)e^{-\Delta V/U_T} - I_{bias} \end{cases} \Longrightarrow \begin{cases} \tau_1\frac{dy_1}{dt} = x_1 - e^{\kappa y}e^{\sigma y_1} \\ \tau_1\frac{dy_2}{dt} = x_2 - e^{\kappa y}e^{\sigma y_2} \\ \tau\frac{dy}{dt} = (e^{\kappa y_1} + e^{\kappa y_2})e^{-y} - 1. \end{cases} \quad (1)$$

by scaling the input signals by the bias currents, $I_1 = x_1 I_{s01}, I_2 = x_2 I_{s01}, I_{s0} = I_{bias}$, and scaling the voltages by U_T, $\Delta V_1 = y_1 U_T, \Delta V_2 = y_2 U_T, V = y U_T$, and defining

$\tau_1 = \frac{CU_T}{I_{s01}}, \tau = \frac{CU_T}{I_{s0}}$, κ as the gate to surface potential coupling for the MOSFET transistor, & σ is the coupling of the drain voltage into the source-side surface potential (assume 0.01 for this example). I_{bias} is the current source current set by V_τ on the gate of Mτ. Transistors M3 and M4, that have the corresponding output currents of I_{out1} and I_{out2}, have their drains at the upper power supply for this example, resulting in both transistors (M3, M4) operating in saturation with subthreshold currents. The capacitors, C, could be either parasitic or explicit capacitances; for clarity of this discussion, all capacitors are considered equal to C. The difference between τ_1 and τ results in a stiff ODE, resulting in higher sampling for stable performance in **Z**-based computation, resulting in a significant computational issue (e.g., [16]), as the I_{bias} and input currents I_1, I_2 can be orders of magnitude different from each other.

Even beyond the significant issue of widely different timeconstants creating a stiff ODE system, these ODEs have competing exponential functions that converge in the physical system but cause tremendous stress on **Z**-based simulation. To eliminate the stiff ODE computation, $I_{s0} = I_{s01}$ resulting in $\tau_1 = \tau$. The resulting ODE system where time is normalized by τ (time becomes unitless):

$$\begin{cases} \dot{y}_1 = x_1 - e^{y} e^{\sigma y_1} \\ \dot{y}_2 = x_2 - e^{y} e^{\sigma y_2} \\ \dot{y} = (e^{y_1} + e^{y_2}) e^{-y} - 1 \end{cases} \rightarrow x_1 = x_2 = 2, \text{ Steady State : } \begin{cases} y_1 = 0 \\ y_2 = 0 \\ y = \ln 2 \end{cases} \tag{2}$$

assuming $\kappa = 1$ for ease of the discussion. One could transform (2) for better numerics, and yet, in generally such a transformation is not possible, so we want to illustrate computational issues with this system The dynamics for (2) for an input step from an initial condition to $x_1 = x_2 = 2$ are analytically globally stable, while a numerical simulation (e.g., RK45) will not always converge to the correct steady-state, including some unstable trajectories (Figure 5).

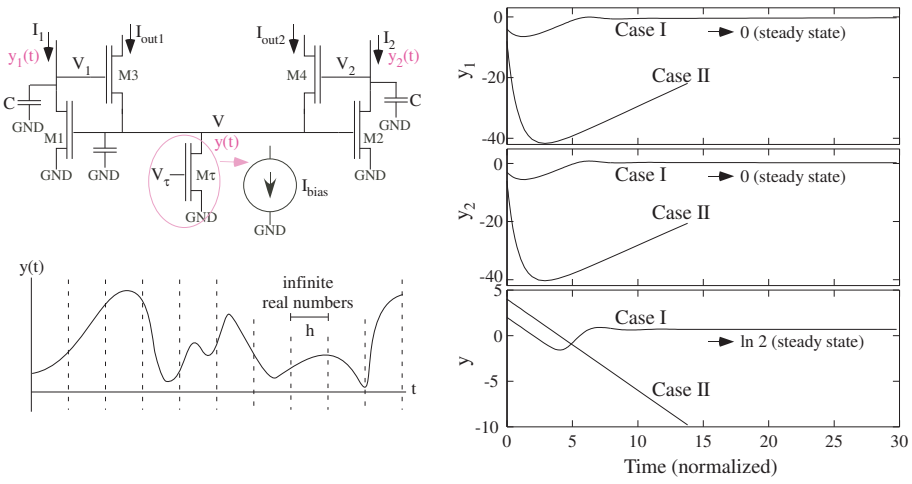

Figure 5. Two-input Winner-Take-All (WTA) circuit dynamics are set through competing nonlinear functions described by y_1, y_2, and y. Time is normalized by τ that is a function of the bias current (I_{bias}) and C. When simulating ODEs using **Z**-valued computation, these competing exponential functions result in large derivatives, resulting in significant errors in the computation, in addition to linear timeconstants that are potentially a different order of magnitude. **R**-valued computation benefits from the **R**-valued timesteps verses **Z**-valued timesteps available for **Z**-valued simulation, as well as **R**-valued computation does not have accumulation errors typical of **Z**-valued integration [16]. In a **Z**-valued timesteps (h) are an ∞ of **R**-valued points. The numerical solution (e.g., RK45) of this two-input WTA circuit results in different stability for a small change in starting conditions. ODE numerical solution for a step from an initial starting condition to $x_1 = x_2 = 2$ for two initial condition cases (Case I: $[y_1\ y_2\ y]$ = [−4 −3 2], Case II: $[y_1\ y_2\ y]$ = [−8 −6 4]). Case I converged, while Case II was numerically unstable.

3. The Continuous-Valued Physical World

The physical world potentially has many opportunities for utilizing continuous-valued functions, including time, space, and amplitude. Computation would be moving from a continuous set of variables, say between 0 and 1, to another function, f(·), say to another continuous set of variables, say between 0 and 1. Analog computation might represent this 0 and 1 as voltages between 0.8 V and 2.2 V, or some other voltage range. On the other hand, the lore in some scientific disciplines strongly states the discrete nature of the physical world. Some authors have strongly endorsed this position (e.g., [39]), although no physical or mathematical foundation is given for these beliefs. Aspects of mathematics are formulated to build continuous functions (e.g., derivatives) to have some validity in a discrete space. Before developing theory for computing over real variables arising from continuous-valued physical representations, these misconceptions about the discrete nature of the physical world needs to be addressed.

*3.1. Noise Does Not Negate **R**-Valued Computation*

Resolution (Figure 6) is a useful abstraction to represent continuous-valued functions with uncertainty and noise, and yet, the abstraction does not constrain the continuous-valued reality. Noise and uncertainty are aspects of the underlying numerical analysis and the resulting computation errors (whether they grow or decrease), but the computation model still operates over real values. The view that real world always has to be interpreted at a finite resolution, through an Analog-to-Digital Converter (ADC) of a finite number of bits, misses the richness of the underlying capability. ADC is a classification between a real-valued world and a discrete-valued world.

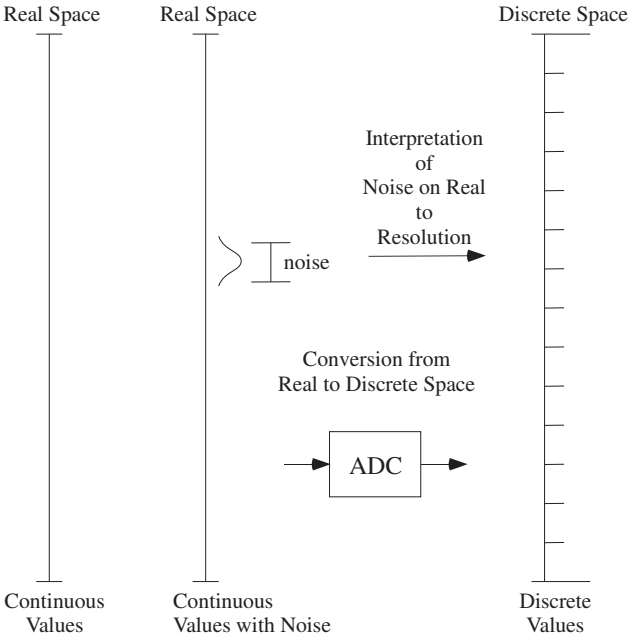

Figure 6. Do physical systems operate over continuous or discrete spaces? Difference between continuous-values with noise versus resolution. One abstraction for real variables with noise and uncertainty is to quantize the number of levels to the extent of that uncertainty. Resolution is this abstraction. In analog systems, this abstraction is computed using an Analog-to-Digital Converter (ADC). This abstraction, while useful for some engineering systems, only approximates the real world, and in no way precludes the continuous-value nature of the resulting computation. The uncertainty would be part of the resulting numerical analysis of the physical computation.

*3.2. The Physical World Is **R**-Valued in Space and Time*

Particles are a useful abstraction to represent continuous-valued physical phenomena, and yet, the abstraction does not constrain the continuous-valued reality. Our abstraction

of physical particles does not preclude the reality of continuous-values in the physical world (Figure 7a). A misconception is that electrons are finite, like marbles, and therefore, in the end, everything is discrete and countable, like a digital Turing machine. Quantum mechanics understands the dual particle and wave nature of reality, although the particle formulation tends to dominate academic thinking in this space [40]. Electron particles can be considered eigenfunctions of the resulting waveform [40]. Furthermore, even if particles were discrete, their position at a particular moment in space and time is a real quantity. Sometimes considering an uncertainty principle, $\Delta x \Delta p > \hbar$, which is similar to time–frequency behavior in signals, as well as $\Delta E \Delta t > \hbar$, might give the sense that the physical world is discrete. The formulation does not constrain the physical world to be discrete, as opposed to continuous, but rather the specificity one gets from a single signal in both areas (x, p) is bounded. Quantum mechanics can restrict or confine a particle based on a particular potential wavefunction, but this confinement, typically over only a couple of variables in a restricted space, is the solution of continuous-functions in amplitude, space, and time. Even if energy is quantitized, position or momentum is not quantitized in all dimensions. Finite measurement capability does not limit the continuous-variable computation.

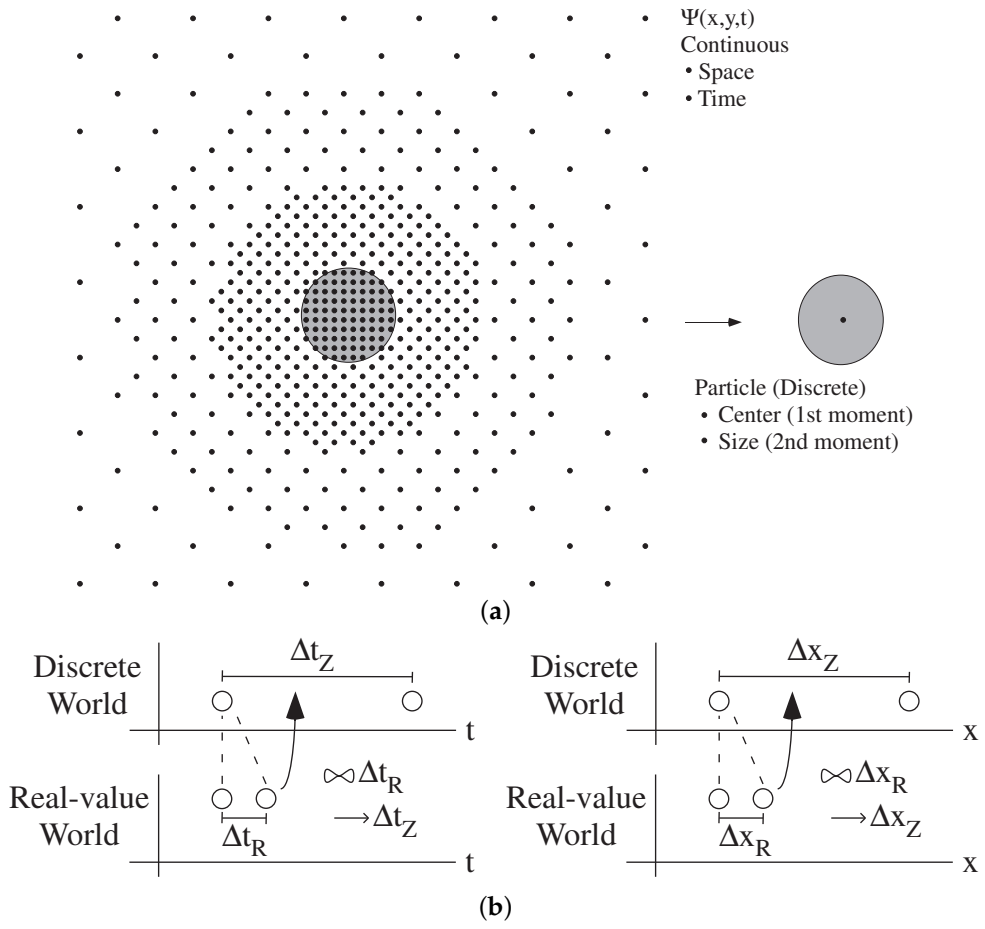

Figure 7. Do particles indicate nature operates over discrete quantities? (**a**) The abstraction of a particle, rather than a wave, effectively takes a first and second moment (mean and approximate extent) of the resulting wave and resulting wavefunction. The position of the particle in space and its position in time are continuous values. (**b**) The computational implication of operating in a **R**-valued world over a **Z**-valued world allows an infinitely more time, space, and amplitude **R**-valued steps between the minimum **Z**-valued step.

Although there is considerable debate between the wave and particle nature of quantum physics (e.g., [40]), the wave nature of quantum mechanics is hard to dispute, and its

resulting continuous variable formulation. Considerable connections have been made to Heisenberg's uncertainty principle and the uncertainty of time and frequency in signal processing [41–43], providing signal processing intuition towards these directions and creating natural bridges between other physical forms of computation. Nothing in this formulation indicates we have a world operating at countable positions in space and time; where discrete particles exist from eigenfunctions of the wave equation, they still live in real valued space and time. **R**-valued space and time gives infinitely more steps, and therefore resources, between each **Z**-value (Figure 7b).

4. Connecting Quantum Computing and Analog Computing Applications

This section shows an example bridging quantum and analog computing in showing an analog circuit model for quantum qubit computation (Figure 8). Quantum computing has been argued that could be performed through analog computing [44–46], have hypothesized parallels with op-amp circuits [47], that **Z**-valued algorithms cannot fully simulate a quantum computer [48], and an initial demonstration through a discrete benchtop analog circuit for a small quantum system (q-bits) utilizing sinusoidal input and output signals [49–51]. Typical quantum computing tends to be performed using fixed devices, such as qubits, and assume that the computation is instantaneous, effectively reaching its steady-state rapidly in the measurement timescales. Therefore, the transformation between **Z**-valued inputs to the measured outputs through these fixed-position qubits, **R**-valued computation described through a Unitary matrix and nonlinear measurement operations. With increased CMOS scaling, analog integrated circuit techniques use more quantum concepts in their fundamental devices, providing another bridge between these techniques.

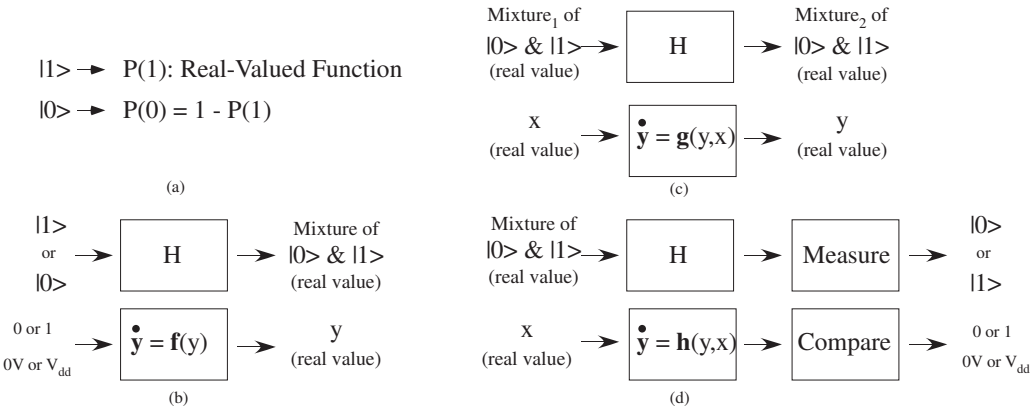

Figure 8. Translation between Qubit-based quantum computation and analog computation. Both Physical computing approaches are computations over real values. (**a**) Although one starts with individual states (<0 or <1), typically one has simultaneous probability of <0 and a probability of <1 (superposition). (**b**) Discrete valued inputs, represented as two quantum or voltage states, through a transformation (**c**) Real-valued inputs, represented as a mixture of two quantum states or an analog voltage, moves through a transformation to another mixture of two quantum states or an analog voltage. (**d**) Real-valued inputs, represented as a mixture of two quantum states or an analog voltage, moves through a transformation and then moves to a discrete value through measurement. For quantum computation, this measure operation typically is the basic nonlinear operation element. This measurement operation for analog computation is a comparison operation of some form.

Quantum computing is primarily a linear computation over the probability wavefunction (Ψ). A typical Quantum computation shows the comparisons between the two physical computations. Although a quantum computation has a single input state (e.g., $|1>$ or $|0>$), the computation involves the combination of these states (Figure 8)

$$|\Psi> \; = \; a\,|1> + b\,|0>,$$

where a, b are complex numbers (=2 real numbers) representing the probability of each state ($|a|^2 + |b|^2 = 1$). The input could also be a combination of these states (Figure 8). The superposition of these two states effectively creates a real-valued representation between 0 and 1, a similar representation having a voltage between 0 V and 1 V.

Multiple quantum operations result in multiple layers of linear operations over complex values. Multiple linear operations can be consolidated to a single linear operation. A single qubit is performing a linear algebraic operation over complex values (Figure 9) that is described through a unitary transformation. A network of qubit operations without measurement results in a single linear algebraic operation that is equivalent to a complex VMM operation described by a unitary matrix. A unitary linear transformation means the total output solution signal power is the same as the total input signal power. The inputs representing the input wavefunction for each initial qubit input. If the input signal power is normalized (to 1) as expected for complex probabilities, then the output signal power is a normalized (to 1) set of complex probabilities. A DFT or DCT or DST or Hadamard transform are all unitary matrix transformations over real or complex values.

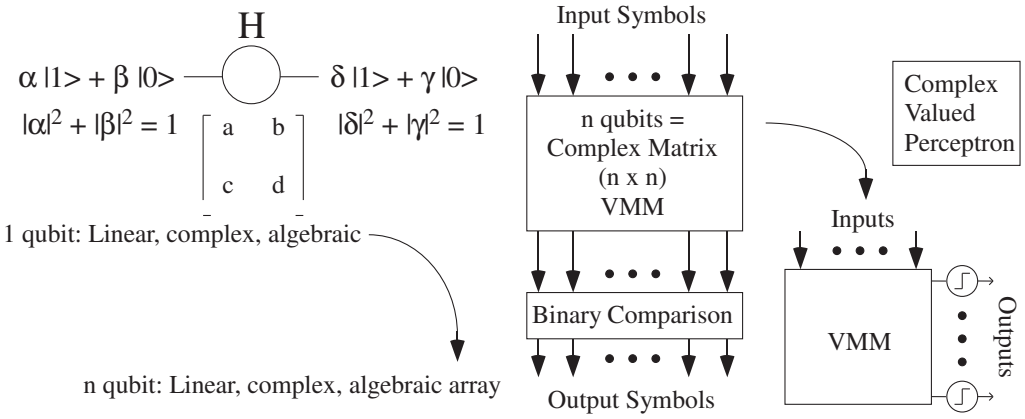

Figure 9. The operation of one or several qubits is a linear operation over complex numbers. Measurement provides a non-linear operation effectively thresholding the resulting probability. A combination of linear, non-time dependent measurements results in a complex VMM where the measurement operation looks like a threshold or a noisy threshold operation. This structure is related to a one-layer Perceptron network, directly implementable by analog computation, with complex weights and real or complex inputs.

An analog VMM (e.g., [8,9]) using complex inputs and complex weights directly computes this formulation (Figures 10 and 11) The rectangular-coordinate complex multiplication of a complex input ($R + j\,I$) and complex weight ($W_R + j\,W_I$) is

$$(R + jI)(W_R + jW_I) = RW_R - IW_I + j(IW_R + RW_I) \tag{3}$$

resulting in four real-valued multiplications (Figure 10), but otherwise resulting in a typical VMM operation. The time-evolution of a qubit array of qubits is typically not considered as the solution is thought to happen instantaneously, often as one sees an analog computation occurs instantaneously. Even so, one can model the linear complex wavefunction dynamics (e.g., Schrodinger's equation) of these devices converging to their steady-state solution through element time-constants (τ) similarly to modeling an analog VMM converging to its linear dynamics through element time-constants (τ). A time domain model could include either feedforward or feedback where the steady-state solution would still simplify to a single complex VMM operation. Using sinusoids of a single frequency with different amplitude and phase (a phasor) provides a second method of implementing the complex operations. The phasor approach computes Hilbert spaces by analog circuits [50,52], theoretically that can be extended to transformations with classical cochlear modeling (e.g., [53,54]), with no difference in overall computational capability.

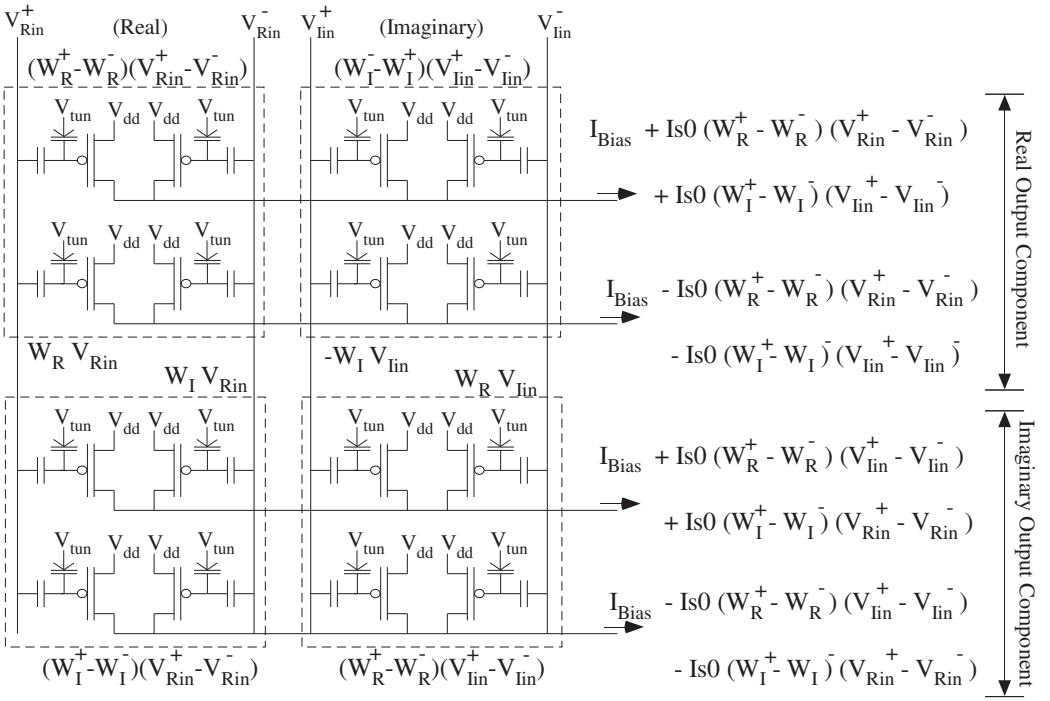

Figure 10. Floating-Gate (FG) based circuit for a complex signal ($V_{inR} + j\,V_{inI}$) multiplied by a complex weight ($W_R + j\,W_I$). The differential complex input terms (V^+_{inR}, V^-_{inR}, V^+_{inI}, V^-_{inI}) to differential complex output terms (Real and Imaginary Output Components), require several partial products to compute the full four-quadrant complex multiplication. V_{tun} is used to erase the FG elements through the tunneling capacitors, and otherwise is held at a fixed potential through this operation.

Superposition is a property of all linear systems (**R** or **Z**) that allows for a number of signals or waveforms to simultaneously exist in an overlapping space. Superposition is taught from the first analog circuit analysis class. Superposition enables simultanious quantum wavefunctions or analog states including for these examples.

Quantum Measurement provides a nonlinear operation to the otherwise linear quantum operations. A measurement makes the wavefunction around that point have a certain value for the duration of the confident measurement. This nonlinearity is a comparison threshold, or a multilevel comparison threshold, with a probability directly related to the magnitude or signal power magnitude of that region's wavefunction. Adding the square of the real and imaginary complex VMM outputs gives this signal power magnitude. The magnitude function operating around a bias current (I_{s0}), uses the differential VMM outputs signals (Figure 10)

$$\text{real} : I^+_{R1} = I_{s0}(1 + X_R)/2, I^-_{R1} = I_{s0}(1 - X_R)/2)$$

$$\text{imaginary} : I_{I1} + I_{s0}(1 + X_I)/2, I^-_{I1} = I_{s0}(1 - X_i)/2)$$

through translinear relationships for transistors with gate voltage changes (e.g., ΔV_1) around the bias current (I_{s0}): [55,56]

$$I_1 = I_{so}e^{\kappa\Delta V_1/2U_T}, I_2 = I_{so}e^{\kappa\Delta V_2/2U_T}$$

$$I_{out} = I_{s0}\left(e^{\kappa V_1/U_T} + e^{\kappa V_2/U_T}\right) = \frac{I_1^2 + I_2^2}{I_{s0}}$$

$$|\Psi|^2 = I_{s0}\Big(((1 + X_R)/2)^2 + (1 - X_R)/2)^2 + ((1 + X_I)/2)^2 + (1 - X_I)/2)^2\Big)$$

$$= I_{s0}\Big(2 + (X_R/2)^2(X_R/2)^2\Big) \quad (4)$$

to create the wavefunction (Ψ) signal power magnitude (Figure 11).

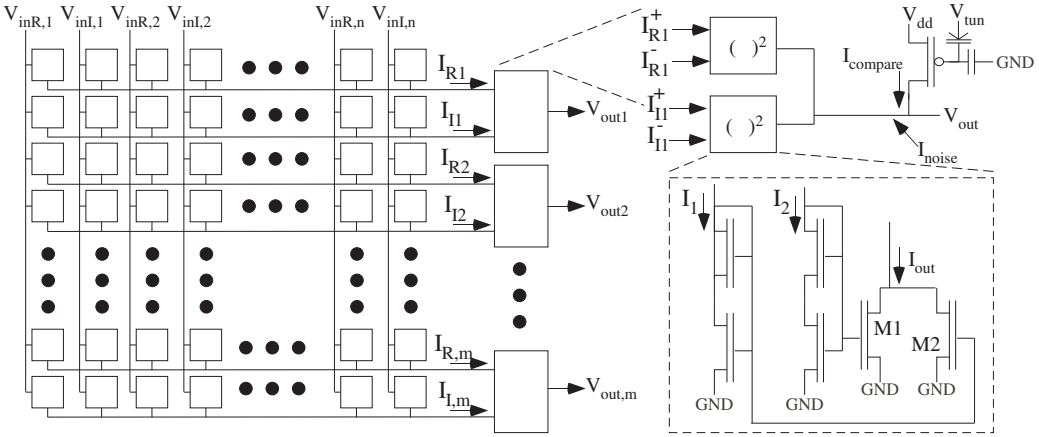

Figure 11. Potential Continuous-Time Computation Architecture for Qubit computation. Source of M1, M2 might be biased higher to keep everything in subthreshold. The complex VMM operation could be either gate or source coupled structure.

The nonlinear measurement operation is a noisy comparison operation to an integer value (e.g., 0 or 1) based on the signal power magnitude of the local wavefunction (e.g., [57]). The square root normalization is not required when passing through a threshold operation or a noisy threshold operation. Thresholds can be set by programming a Floating-Gate (FG) transistor at the threshold current. An analog noise generator (e.g., [13]) before the comparison gives the probabilistic output related to the local wavefunction magnitude, as needed, beyond the comparator circuit noise. Some algorithms may not benefit from additional noise before the comparison. The probability may be represented by a dithering of the output, similar to rate-encoded values seen in integrate and fire neurons, effectively looking like a noise source. Noisy dithered outputs for Quantum computing enables Quantum systems to be efficient noise generators (e.g., [11]), and likely was the reason the first demonstration of physical supremacy (called Quantum supremacy) for Quantum computing was a complicated noise source.

As a result of these properties, quantum computing is a form of perceptron computation over complex-valued quantities, where the input and output results are discrete elements (Figure 9). This structure is analogous to an analog implementation (Figure 11) of a one-layer Perceptron network with complex weights and real or complex inputs (Figure 9). Multivalued symbol output can be abstracted where the equivalent of more outputs states is encoded into having a factor of more qubit outputs (e.g., 4 symbols → two outputs, 8 symbols → three outputs). Note that quantum entanglement is not part of these computations, as entanglement is about quantum error correction and not primary computations. Shor's algorithm [35] directly solved through this analog, room-temperature equivalent circuit that is similar to analog Fourier transforms (e.g., [58]). An engineer could still choose either method for solution depending on other engineering constraints.

5. Relationships between Physical Computing Approaches

Every form of **R**-valued computing has equivalents to other **R**-valued techniques. These concepts build on the existence and properties of a continuous-valued environment. Optical, Neuromorphic, Quantum, and Analog computing compute over **R** amplitude and **R** time with various spatial forms (Figure 12). All four physical systems described show properties of superposition within their linear operating region, allowing for a number of signals or waveforms to simultaneously exist in the same representation. Optical communication systems extensively make use of superposition to have multiple frequency or wavelength carriers communicate on the same fiber. All four physical systems may have infinite time and/or spatial responses. Optimization problems are routinely solved in

analog, quantum and neuromorphic techniques, resulting from coupled ODEs or PDEs propagating energy down the established energy surfaces, either to a global minimum or local minimas. Quantum computing relaxation and annealing as in Grover's algorithm [36], find similar concepts within as well as analog energy relaxation techniques (e.g., [59]) and neuromorphic L_1 norm minimization [60]. A good recent review shows the different physical computing techniques for similar energy and power surface minimization [61].

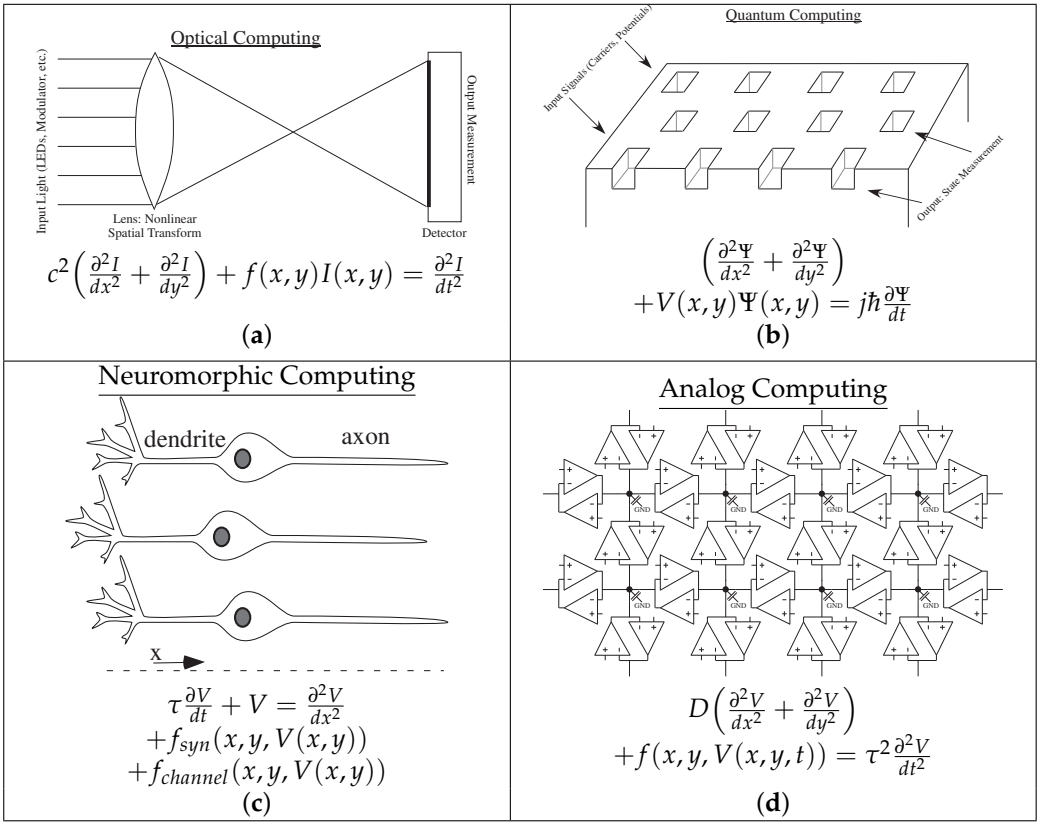

Figure 12. Example computational medium for Physical Computing. All cases are described by linear or nonlinear PDEs capable of diffusion and wave propagation, where they operate over space as well as one continuous variable as well as continuous time. We show representative PDEs for potential computations for each case; the diagrams do not capture all possible phenomena. (**a**) Optical Computing: Highly efficient and linear computation using light in multiple spatial dimensions, described through second-order wave equations. (**b**) Quantum Computing: Wave-based physics coupling particles to all other particles. Typical implementations tend to use multiple quantum wells, energy barriers (modeled by V), and connected states implementing computation through the resulting wavefunctions (Ψ). (**c**) Neuromorphic Computing: Physical computing inspired by computing in animal nervous systems. Looking along a single axis of a single neuron, the physics of the membrane voltage (V) is a combination of diffusion and waveguiding behavior. (**d**) Analog Computing: An example utilizing analog devices computing in voltage (V) a temporal-spatial PDE including second-order space and time dynamics. These systems have some continuous spatial behavior through the spatial dynamics in individual dynamics, whether or not those intermediate results are used.

Second-Order Partial Differential Equations (PDE) provide a translatable framework between these different **R**-valued computational mediums (Figure 12) computing over **R**-valued amplitude, space, and time (Table 1). Optical computing utilizes waves, described through a second-order time and space PDE formulation of Maxwell's equations, computing through continuous time, amplitude, and 1- to 3-dimensional space utilizing a number of spatial filters, lenses, and a variety of tunable light modulators & mirrors can modify an input optical signal. Quantum computing uses complex probability wavefunctions (Ψ)

governed through PDEs (Schrodinger's equation) computing through continuous time, amplitude, and 1- to 3-dimensional space. Typical implementations compute multiple quantum wells, potential energy barriers, and connected states governed by these PDEs. Neuromorphic computing (e.g., [10,62]), including Neural Networks (e.g., [30,31]), uses physical computing devices, typically of an analog nature in Si or hybrid system, modeling part of a neurobiological computations (e.g., neurons) using continuous amplitude, time, with at least 1-dimensional space PDEs in neuron computation (e.g., dendritic systems) [63]. Neurons are spatio-temporal computing elements with hundreds if not thousands of inputs, modeling voltages governed by diffusing and wave-propagating PDEs (Figure 12). Analog neuron implementations have demonstrated hundreds of inputs (e.g., [64]) as well as wave-propagating PDEs (e.g., [63]). The dendritic PDEs have a significant linear operating range within the overall biological structure.

Table 1. Comparison between **R**-valued computing techniques.

	Time	Amplitude	Space	R Values	Core PDEs
Quantum	1	1	1–3	>=3	1st/2nd-Order Hyperbolic
Analog	1	1	0–3?	>=2	1st & 2nd-Order Space & Time
Neuromorphic	1	1	1	>=3	2nd-Order Parabolic
Optical	1	1	1–3	<=3	2nd-Order Hyperbolic

Analog computing operates in continuous-time and amplitude within a spatially coupled environment and described by multiple coupled differential equations, including coupled linear or nonlinear PDEs utilizing first and second order space and time dynamics (e.g., [65]). The physical system is **R**-value in space, but practically the parameters change in particular points with a finite granularity of parameter resolution setting, as well as output measurement capability. The PDE could be coupled transistors (e.g., resistive networks [59]) or transistor circuits (e.g., ladder filters [65]). Inputs, outputs and boundary conditions are set through additional analog circuitry. Analog techniques provide the mode advanced physical implementation capabilities, including programmable and reconfigurable techniques in standard CMOS processes [18].

A bridge between computing capabilities results in a win-win silicoation (Figure 4), where results developed from one physical system can be translated to a second physical system. Although one medium is more efficient than another for particular problems, one can transform between the two mediums through polynomial-size transformations. For example, analog, optical, quantum, and neuromorphic computing can compute VMM operations. Lenses, programmable modulators, and programmed micromirrors enable optical VMM computations. Analog and Quantum operations were discussed in the previous section. Neuromorphic operations are likely the most nonlinear of these approaches.

The paths between these systems starts by translating the core PDEs between each system (Figure 12), although the process might be simplier when computing concepts use only a part of the physical medium's capability (e.g., Section 4). An earlier example showed moving between a quantum system (qubits) and an analog computing system, one of the more challanging **R**-valued translations (Section 4). The translation between optical and quantum computing goes through their similar hyperbolic wave-propagating PDEs and similar mathematical formulations, where both heavily utilize the temporal and spatial duality between real values and Fourier transformed values. Analog techniques involve a wide range of ODE and PDE techniques that include the range of linear and nonlinear PDE systems. Analog techniques can compute the same PDEs for Maxwell's equations as optical systems (e.g., [65]), although with a polynomially larger complexity in many cases. Neuromorphic systems utilize parabolic PDEs (e.g., diffusion), and yet these networks can be approximate waveguiding systems by altering spatial parameters [63] as well as through local (e.g., active channels) or active network (e.g., synfire chain [66]) properties. The translation between analog and neuromorphic is straight-forward as most

neurormophic models are built with some analog circuit modeling (e.g., [10,62]). The spatial steady-state solutions typically form elliptic PDE problems (Poison's equation). Each approach has a linear region, although nonlinear operations are harder in some domains (e.g., Optical), possible in some cases (e.g., Quantum), and easy or too easy in other cases (e.g., Analog or Neuromorphic). In each case, even and odd nonlinearities are possible and a linear region are possible where some systems are more optimal in some places than others. One expects translations between other physical systems not identified, all systems computing with multiple **R**-valued operations and representations.

6. Computing Opportunities for Physical Computation

After defining physical computing, describing its properties (**R** versus **Z**), as well as showing the transformations between these techniques, the discussion moves to considering the opportunities for physical computing. At one level, physical computing approaches often have far higher computational efficiency and lower computational area [7–9,18,19], as mentioned for analog systems ($1000\times$, $100\times$) at the beginning of this discussion. Neuromorphic approaches have demonstrated even greater computational efficiencies with roadmaps possible for both improved analog and neuromorphic cases [10]. These opportunities alone are sufficient to explore these techniques, particularly in an ever energy-constrained environment. The availability of large-scale programmable and configurable analog techniques has enabled a realistic analysis of physical computing techniques.

Some examples in P_A improve P_D algorithms, although remain in P_A (Figure 13). Spaghetti sort [67] solved using analog WTA networks [14] in O(M) time rather than greater than O(Mlog M) time for physical digital implementations. Grover's quantum computing algorithm also improved the O(·) time for the related algorithms [36].

Sometimes these physical computing discussions involve energy relaxation down an energy surface to a minimum level; a good review of these physical computing techniques can be found elsewhere [61]. These approaches map well to physical computing approaches, and with just the right formulation or resulting energy surface, one could have a single global minimum, resulting in an optimal solution. In general, there are local minimas that might represent good solutions, and therefore these physical computing techniques do not result in a P_A solution for NP_D problems (Figure 13), although they might provide good solutions for many different cases. Analog resistive networks can create parabolic PDE with elliptical PDE steady states to create energy surfaces, energy surfaces similar to soap bubbles around fixed boundary conditions. These systems provide good solutions, but local minimas tend to limit the extent of optimality, typical of these approaches in any medium. For example, using resistive networks often achieve 96% correct for optimal path planning, but not full optimality [59]. Very good solutions have been seen using energy surface relaxations for NP_D problems (e.g., TSP) using physical techniques [68,69]. All well designed energy surface relaxation or related methods implemented in physical systems show similar results. These perspectives often lead individuals to believe that physical computing will not solve NP_D problems (e.g., Quantum [48]). Unless just the right surface is found and implemented, simple relaxation does not indicate whether NP_D problems can be solved in P_A time.

At another level, computing over **R** versus **Z** likely has additional benefits including addressing the relationship between NP_D and $EXPTIME_D$ to P_A and NP_A (Figure 13). Some examples in P_A appear to be beyond P_D. such as Shor's quantum factorization algorithm [35]. Multiple coupled ODEs systems have been proposed to solve NP problems, such at the 3-SAT problem [33,34,70–72], that could be implemented on a configurable continuous-time platform [19], although to date it has not been verified through physical computing approaches. The potential overlap of $EXPTIME_D$ (e.g., Shor Factorization) with P_A and NP_A gives motivation for deeper explorations. Throughout this process, synchronous digital simulation results of physical computation must always be approached cautiously, as digital computation is more limited than the resulting physical substrate (Figure 4).

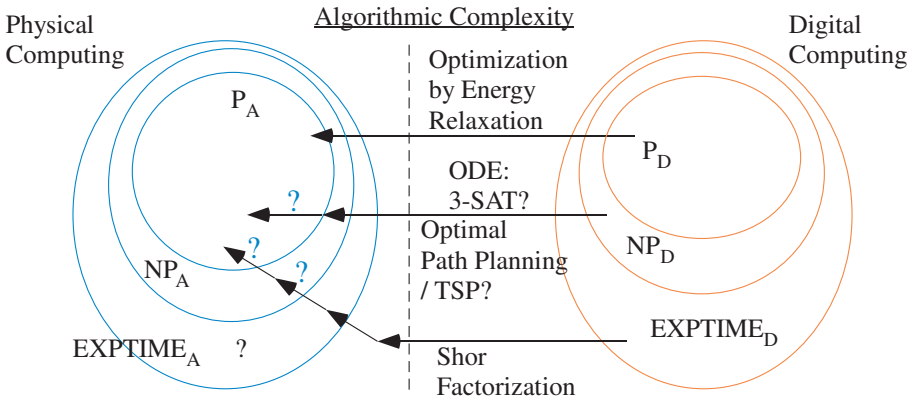

Figure 13. Mapping the transformation between **Z** computations and **R** computations remains an open question. Some algorithms that solve in P_D, such as energy surface relaxation applications, directly translate to algorithms in P_A, even if they sometimes give good enough solutions for NP_D, NP_A. Some algorithms might provide an opportunity to bridge between NP_D and P_A.

Optimal path planning is computed in a polynomial size array of neurons in polynomial time [66], providing one approach to connect NP_D and P_A (Figure 13). Physically based wave-propagation computation using neural events is proven analytically and experimentally to give an optimal solution [66], while energy surface relaxation nearly, but not always (96% correct [59]) reaches the optimal solution. These solutions enable an optimal solution by using wave propagation, rather than diffusive solutions falling down energy surfaces. The optimal solution is analogous to timing of optical waves through a set of barriers along the path. The first path that arrives is the winning path, typical of optical systems. Algorithm intuition sometimes arises by equating one physical solution to another physical solution. The principle of least time optimization [61] is the closest connection to this physical optimal path planning through active propagation [66]. This physical implementation is related to Dijkystra's graph optimization algorithm, but with strong physical optimality results. Dijkstra's algorithm computes an optimal path over a graph that may have been abstracted from a physical map within the timestep resolution.

Optimal path computation over a graph within the parameters set for the Dijkystra algorithm is optimal, and that particular problem is not an NP computation. Sometimes multiple paths are considered equal within the graph step timing resolution, even though one path is shorter. Continuous-time physical computation eliminates that ambiguity, only limiting the output computation to the final output readout resolution, creating a stronger definition of optimality. The class of problems for this stronger definition of optimality is unclear, requiring arbitrary smaller and smaller timesteps. The continuous variable allows for arbitrary timing resolution because the resulting computation is over continuous variables. If NP_D is demonstrated part of P_A in one physical computing platform, then it is true in all fully-capable **R**-valued computing platforms. It remains an open question to connect the Traveling salesman Problem (TSP) or Max cut problem to this **R**-valued optimal path planning. This discussion sits at a place where we have the framework to explore if NP_D is part of P_A, looking for a constructive approach based on some potential examples.

7. Summary and Discussion: Implications and Opportunities of Physical Computation

We presented a computing model for physical computing, computing over real values including analog, neuormorphic, optical, and quantum computing. These techniques build upon recent innovations in analog/mixed-signal computation as well as analog modeling, architectures and algorithms. This approach shows a potential physical computing model, enabling similar capabilities to digital computation with its deep theoretical grounding starting from Turing Machines. This paper addressed the possibility of a real-valued or analog Turing machine model, and the potential computational and algorithmic opportunities. These techniques have shown promise to increase energy efficiency, enable

smaller area per computation, and potentially improve algorithm scaling. This effort describes opportunities beyond coefficient or minor polynomial scaling improvements in computational energy efficiency, but rather could significantly improve the computational energy efficiency scaling, $O(\cdot)$, resulting from fundamental algorithmic improvements from computing over **R**-values as compared to traditional digital computing over **Z**-values.

Although there is a starting theory of **R**-valued computation and its **R**-valued Turing Machine, and there is starting theory of **R**-valued numerical analysis, architectures, and abstraction primarily coming from analog implementations, one still requires a constructive framework for engineers to design physical computing systems for an application. Although there are significant guideposts for physical computation so we no longer require a miracle to occur (Figure 1) for such implementations, one still does not find the well-traveled paths typical of linear digital design.

A constructive framework for physical computing design requires a constructive design approach using nonlinear dynamics. Engineers are well versed in designing signal processing and control systems using linear dynamical systems and differential equations. If any part of the system is nonlinear, or perceived as nonlinear, the design effort, risk, and stress becomes almost unmanagable as there are few tools for designing nonlinear systems. Although theoretical analytical tools for nonlinear dynamics is extensive (e.g., [73]), engineers have nearly no resources to design with nonlinear systems, resulting in another a miracle to occur gap.

As an example in analog circuits, silicon nonlinearities are $\tanh(\cdot)$, $\sinh(\cdot)$, $e^{(\cdot)}-1$ as well as their inverses, providing the even (x^2) and odd (x^3) normal forms [73,74]. Handling these nonlinearities requires the right current (I) and voltage (V) relationships. For example, the output of a VMM is a current, so a compressive odd nonlinearity following this operation would not be a $\tanh(\cdot)$ ($I=\tanh(V)$), but could be a $\sinh^{-1}(\cdot)$ ($I=\sinh(V)$). Frameworks for solving linear equations [75] adapted to allow for the natural nonlinearities could generate fairly general sets of coupled differential equations with even and odd nonlinearities. The analog form of L1 minimization is similar to these circuit structures [60].

A constructive framework for physical computing design requires a constructive design approach for applying physical algorithms towards an application. Often several energy-space algorithms with the hope they will provide a a good to optimal solution for a particular application. Most of the theory for choosing these algorithms is hueristic in nature, effectively resulting from the artistry of the expert designer. Other algorithmic approaches are also possible, and yet further development in physical computing platforms is essential. For example, can neural path planning algorithm [66] be modified to solve NP-class applications (e.g., TSP, Max-Cut, or 3-SAT)? As many transformative problems will experience chaotic dynamics in their solution (e.g., [33,34]), and as a result will become unfeasible to solve in PD, understanding chaotic analog dynamics, as well as implementing these dynamics in a physical system (e.g., FPAA) provides a guide for algorithm development. Because of the lack of such structures, nonlinear dynamics has hitherto not exploited these larger opportunities (e.g., [33,34]).

These remaining open questions of constructive frameworks for designing with nonlinear dynamics, choosing the right algorithms as well as choosing the right physical medium for a particular application, are significant challenges going forward. Furthermore, yet, now being able to define these challenges carefully within a clear framework, working through these challenges creates the opportunity of well-traveled paths through physical computing. Educating and creating a community of engineers with these new tools becomes essential towards solving and utilizing these spaces.

Author Contributions: J.H.: Formal analysis, writing original draft, figures. E.B.: writing original draft. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable for studies not involving human or animals.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Turing, R. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **1937**, *2*, 230–265.
2. Moore, G.E. Cramming more components onto integrated circuits. *Electronics* **1965**, *38*, 114–117.
3. Moore, G.E. Progress in Digital Integrated Electronics. *IEEE IEDM Electron Devices Meet.* **1975**, *21*, 11–13.
4. Hoeneisen, B.; Mead, C.A. Fundamental Limitations in Microelectronics—I. MOS Technology. *Solid State Electron.* **1972**, *15*, 819–829.
5. Mead, C.; Conway, L. *Introduction to VLSI System Design*; Addison-Wesley: Boston, MA, USA, 1980; ISBN 0-201-04358-0.
6. Grier, D.A. *When Computers Were Human*; Princeton University Press: Princeton, NJ, USA; Oxford, UK, 2005.
7. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **1990**, *78*, 1629–1636.
8. Chawla, R.; Bandyopadhyay, A.; Srinivasan, V.; Hasler, P. A 531 nW/MHz, 128 × 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity. In Proceedings of the IEEE 2004 Custom Integrated Circuits Conference, Orlando, FL, USA, 6 October 2004; pp. 651–654.
9. Schlottmann, R.C.; Hasler, P.E. A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2012**, *1*, 403–411.
10. Hasler, J.; Marr, H.B. Finding a roadmap to achieve large neuromorphic hardware systems. *Front. Neurosci.* **2013**, *7*, 118.
11. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* **2019**, *574*, 505–510.
12. Koetsier, J. IBM: Google's 'Quantum Supremacy' Claim Is Wrong—150 Million Percent Wrong. *Forbes*, 23 October 2019.
13. Marr, B.; Hasler, J. Compiling Probabilistic, Bio-Inspired Circuits on a Field Programmable Analog Array. *Neuromorphic Eng. Syst. Appl.* **2015**, 88–96, doi:10.3389/fnins.2014.00086.
14. Hasler, J. Analog Architecture and Complexity Theory to Empowering Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems. *J. Low Power Electron. Appl.* **2019**, *9*, 4.
15. Hasler, J. Opportunities in Physical Computing driven by Analog Realization. In Proceedings of the 2016 IEEE International Conference on Rebooting Computing (ICRC), San Diego, CA, USA, 17–19 October 2016.
16. Hasler, J. Starting Framework for Analog Numerical Analysis for Energy Efficient Computing. *J. Low Power Electron. Appl.* **2017**, *7*, 17.
17. Hasler, J.; Kim, S.; Natarajan, A. Enabling Energy-Efficient Physical Computing through Analog Abstraction and IP Reuse. *J. Low Power Electron. Appl.* **2018**, *8*, 47.
18. Hasler, J. Large-Scale Field Programmable Analog Arrays. *IEEE Proc.* **2020**, *108*, 1283–1302.
19. George, S.; Kim, S.; Shah, S.; Hasler, J.; Collins, M.; Adil, F.; Wunderlich, R.; Nease, S.; Ramakrishnan, S. A Programmable and Configurable Mixed-Mode FPAA SoC. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 2253–2261.
20. Collins, M.; Hasler, J.; George, S. An Open-Source Toolset Enabling Analog–Digital Software Codesign. *J. Low Power Electron. Appl.* **2016**, *6*, 3.
21. Hasler, J.; Natarajan, A. An Open-Source ToolSet for FPAA Design. In Proceedings of the WOSET, 2020. Available online: http://hasler.ece.gatech.edu/SoCFPAA/WOSET2020SoCFPAATools.pdf (accessed on 25 March 2021).
22. Siegelmann, H.T.; Fishman, S. Analog computation with dynamical systems. *Physica D* **1998**, *120*, 214–235.
23. Cabessa, J.; Siegelmann, H.T. The Super-Turing Computational Power of Plastic Recurrent Neural Networks. *Int. J. Neural Syst.* **2014**, *24*, 1450029-1–1450029-22.
24. Deutsch, D. Quantum Theory, the Church-Turning Principle and the Universal Quantum Computer. *Proc. R. Soc. Lond. Ser. A Math. Phys. Sci.* **1985**, *400*, 97–117.
25. Feynman, R.P. Simulating Physics with Computers. *Int. J. Theor. Phys.* **1982**, *21*, 467–488.
26. Feynman, R.P. Quantum Mechanical Computers. *Found. Phys.* **1986**, *16*, 507–531.
27. Hopfield, J.J.; Tank, D.W. Neural computation of decisions in optimization problems, *Biol. Cybern.* **1985**, *52*, 141–152.
28. Atiya, A.E.; Abu-Mostafa, Y.S. An analog feedback associative memory. *IEEE Trans. Neural Netw.* **1993**, *4*, 117–126.
29. Abu-Mostafa, Y.S. Information theory, complexity, and neural networks. *IEEE Commun. Mag.* **1989**, *27*, 25–28.
30. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558.
31. Hopfield, J.J. Neurons with graded responses have collective computational properties like those of two- state neurons. *Proc. Natl. Acad. Sci. USA* **1984**, *81*, 3088–3092.
32. Siegelmann, H.T. Turing on Super-Turing and Adaptivity. *Prog. Biophys Mol. Biol.* **2013**, *113*, 117–126.
33. Ercsey-Ravasz, M.; Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nat. Phys* **2011**, *7*, 966–970.
34. Yin, X.; Sedighi, B.; Varga, M.; Ercsey-Ravasz, M.; Toroczkai, Z.; Hu, X.S. Efficient Analog Circuits for Boolean Satisfiability. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *26*, 155–167.
35. Shor, P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994.

36. Grover, L.K. A Fast Quantum Mechanical Algorithm For Database Search. In Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996.
37. Hasler, J.; Shah, S. SoC FPAA Hardware Implementation of a VMM+WTA Embedded Learning Classifier. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 28–37.
38. Butcher, J.C. *Numerical Analysis of Ordinary Differential Equations: Runga Kutta and General Linear Methods*; Wiley: New York, NY, USA, 1987.
39. Wheeler, J.A. Information, Physics, Quantum: The Search for Links. *Inf. Phys.* **1989**, *19*, 309–336.
40. Mead, C.A. *Collective Electrodynamics: Quantum Foundations of Electromagnetism*; MIT Press: Cambridge, MA, USA, 2000.
41. Daugman, J.G. Complete Discrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression. *IEEE Trans. Acoust. Speech Signal Process.* **1988**, *36*, 1169–1179.
42. Daugman, J.G. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A* **1985**, *2*, 1160–1169.
43. Gabor, D. Theory of communication. *J. IEE* **1946**, *93*, 429–457.
44. Ferry, D.K.; Akis, R.; Harris, J. Quantum wave processing. *Superlattices Microstruct.* **2001**, *30*, 81–94.
45. Ferry, D.K.; Akis, R.; Knezevic, I. Quantum waves:the proper basis for low dissipation quantum computing. *Microelectron. Eng.* **2002**, *63*, 17–21.
46. Ferry, D.K.; Akis, R.; Gilbert, M.J.; Knezevic, I. Do we need quantum for quantum computing? In Proceedings of the SPIE 5115—Noise and Information in Nanoelectronics, Sensors, and Standards, Santa Fe, NM, USA, 9 May 2003.
47. Dyson, C.P.P. Implementing Quantum Algorithms Using Classical Electrical Circuits: Deutsch, Deutsch-Jozsa, and Grover. Master's Thesis, University of York, York, UK, 2011.
48. Preskill, J. Quantum Computing in the NISQ era and beyond. *arXiv* **2018**, arXiv:1801.00862v3.
49. La Cour, B.R.; Sudarshan, E.C.G. Classical model for measurements of an entanglement witness. *Phys. Rev. A* **2015**, *92*, 032302-1–032302-7.
50. La Cour, B.R.; Ott, G.E. Signal-based classical emulation of a universal quantum computer. *New J. Phys.* **2015**, *17*, 053017. doi:10.1088/1367-2630/17/5/053017.
51. La Cour, B.R.; Ostrove, C.I.; Ott, G.E.; Starkey, M.J.; Wilson, G.R. Classical emulation of a quantum computer. *Inter. J. Quantum Inf.* **2016**, *14*, 1640004-1–1640004-12.
52. Kish, L.B. Hilbert Space Computing by Analog Circuits. In Proceedings of the SPIE 5115—Noise and Information in Nanoelectronics, Sensors, and Standards, Santa Fe, NM, USA, 9 May 2003.
53. Sarpeshkar, R. Emulation of Quantum and Quantum-Inspired Spectrum Analysis and Superposition with Classical Transconnductor-Capacitor Circuits. U.S. Patent US 10,204,199, 12 February 2019.
54. Sarpeshkar, R. Emulation of Quantum and Quantum-Inspired Discrete-State Systems with Classical Transconductor-Capacitor Circuits. U.S. Patent US 10,275,556 B2, 30 April 2019.
55. Minch, B.A.; Diorio, C.; Hasler, P.; Mead, C. Translinear circuits using subthreshold floating-gate MOS transistors. *Analog Integr. Circuits Signal Process.* **1996**, *9*, 167–179.
56. Minch, B.; Hasler, P.; Diorio, C. Multiple-input translinear element networks. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **2001**, *48*, 20–28.
57. Lanham, S.A.; La Cour, B.R. Detection-Based Measurements for Quantum Emulation Devices. In Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), Denver, CO, USA, 12–16 October 2020; pp. 37–46.
58. Suh, S.; Basu, A.; Schlottmann, C.; Hasler, P.; Barry, J. Low-power discrete Fourier transform for OFDM: A programmable analog approach. *IEEE Trans. Circuits Syst. I* **2011**, *58*, 290–298.
59. Koziol, S.; Wunderlich, R.; Hasler, J.; Stilman, M. Single-Objective Path Planning for Autonomous Robots Using Reconfigurable Analog VLSI. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *47*, 1301–1314.
60. Shapero, S.; Charles, A.S.; Rozell, C.J.; Hasler, P. Low power sparse approximation on reconfigurable analog hardware. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2012**, *2*, 530–541.
61. Vadlamania, S.K.; Xiaob, T.P.; Yablonovitch, E. Physics successfully implements Lagrange multiplier optimization. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 26639–26650.
62. Mead, C. *Analog VLSI and Neural Systems*; Addison Wesley: Reading, MA, USA, 1989.
63. George, S.; Hasler, J.; Koziol, S.; Nease, S.; Ramakrishnan, S. Low-power dendritic computation for wordspotting. *J. Low Power Electron. Appl.* **2013**, *3*, 78–98.
64. Brink, S.; Nease, S.; Hasler, P.; Ramakrishnan, S.; Wunderlich, R.; Basu, A.; Degnan, B. A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst.* **2013**, *7*, 71–81.
65. Hasler, J.; Shah, S. Reconfigurable Analog PDE computation for Baseband and RF Computation. In Proceedings of the GOMAC, Reno, NV, USA, 20–23 March 2017.
66. Koziol, S.; Brink, S.; Hasler, J. A neuromorphic approach to path planning using a reconfigurable neuron array IC. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 2724–2737.
67. Dewdney, A.K. On the spaghetti computer and other analog gadgets for problem solving. *Sci. Am.* **1984**, *250*, 19–26.
68. Feng, G.; Douligeris, C. Using Hopfield Networks to Solve Traveling Salesman Problems Based on Stable State Analysis Technique. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, 27 July 2000; Volume 6.

69. Mandziuk, J. Solving the traveling salesman problem with a Hopfield-type neural network. *Demonstr. Math.* **1996**, *29*, 219–231.
70. Vergis, A.; Steiglitz, K.; Dickinson, B. The Complexity of Analog Computation. *ACM J. Math. Comput. Simul.* **1986**, *28*, 91–113.
71. Cocco, S.; Monasson, R. Analysis of the computational complexity of solving random satisfiability problems using branch and bound search algorithms. *Eur. Phys. J. B* **2001**, *22*, 505.
72. Sumi, R.; Varga, M.; Toroczkai, Z.; Ercsey-Ravasz, M. Order-to-chaos transition in the hardness of random Boolean satisfiability problems. *Phys. Rev. E* **2016**, *93*, 052211.
73. Strogatz, S.H. *Nonlinear Dynamics and Chaos*; John Wiley & Sons: New York, NY, USA, 2015.
74. Ott, E. *Chaos in Dynamical Systems*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2002.
75. Hasler, J.; Natarajan, A. Continuous-time, Configurable Analog Linear System Solutions with Transconductance Amplifiers. *IEEE Trans. Circuits Syst. I* **2021**, *68*, 765–775.