Technical Report

Perturbation-Based Computing for Next-Generation Embedded IT Targeted at Emerging Nanoelectronics

Andrey Zykov, Margarida Jacome, and Gustavo de Veciana
Department of Electrical & Computer Engineering
The University of Texas at Austin

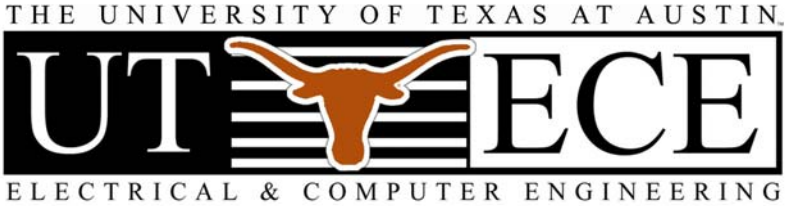
UT-CERC-10-01

January 14, 2010

Computer Engineering Research Center
The University of Texas at Austin
1 University Station, C8800
Austin, Texas 78712-0323

Telephone: 512-471-8000
Fax: 512-471-8967

http://www.cerc.utexas.edu



Perturbation-Based Computing for Next-Generation Embedded IT Targeted at Emerging Nanoelectronics

Andrey Zykov, Margarida Jacome, and Gustavo de Veciana
Department of Electrical & Computer Engineering
The University of Texas at Austin

***Abstract*—This paper makes the case for perturbation-based computational models as a promising choice for implementing next generation ubiquitous information technology on unreliable nanotechnologies. We show the inherent robustness of such computational models to high defect densities and performance uncertainty which, when combined with low manufacturing precision requirements, makes them particularly suitable for emerging nanoelectronics. We propose a hybrid eNano-CMOS perturbation-based computing platform relying on a new style of configurability that exploits the computational model's unique form of unstructured redundancy. We consider the practicality and scalability of perturbation-based computational models by developing and assessing initial foundations for engineering such systems. Specifically, new design and decomposition principles exploiting task specific contextual and temporal scales are proposed and shown to substantially reduce complexity for several benchmark tasks. Our results provide strong evidence for the relevance and potential of this class of computational models when targeted at emerging unreliable nanoelectronics.**

***Index Terms*—Nanotechnology, Real time systems, Signal processing**

I. Introduction

Advances in the synthesis and self-assembly of nanoelectronic devices suggest that the ability to manufacture dense nanofabrics is on the near horizon [1], [2], [3], [4], [5], [6], [7]. Yet, effective ways of utilizing emerging nanoelectronic technologies still elude us. The tremendous increase in device densities afforded by nanotechnologies is expected to be accompanied by substantial increases in defect densities, performance variability, and susceptibility to single event upsets caused by cosmic radiation (energetic neutrons) and alpha particles [2], [8], [5]. System-level design adhering to current computational models may thus soon reach fundamental scaling limits, where the increased densities are countered by overheads associated with achieving defect- and fault-tolerant designs that are robust to performance variability [8], [9], [10], [11]. Thus, it is critical to consider and explore alternative computational models that can operate under such difficult conditions.

Additionally, the nature of next generation ubiquitous information technology (IT) – including many challenging real-time streaming media applications, such as voice and image recognition, as well as a myriad of automation/control and robotics applications – also calls for rethinking current computational models and associated design paradigms. Specifically, in order to enable the massive embedded systems' deployment required by next generation ubiquitous IT, it is imperative to rely on low design cost/complexity platforms that can be easily configured to implement the many tasks at hand, with acceptable performance. Unfortunately, the cost and complexity of system-level design adhering to current computational models continues to increase dramatically, conflicting with these requirements.

Contributions. In this paper, we investigate a promising new class of non-Turing computational models, called perturbation-based, and show its potential to synergistically address the two sides of the complex system design equation: technology and applications. Our argument on the suitability of this computational model for next generation IT systems targeted at nanotechnologies is based on five main points – the first three relate to technology issues while the remaining address system-level design and application issues. Specifically, as will be seen, the suitability of perturbation-based computing for emerging nanoelectronics ('eNano') technologies is predicated on: (1) its reliance on a computational core that can, to a large extent, be 'randomly assembled', thus relaxing strict manufacturing precision and stability requirements; (2) its inherent tolerance to manufacturing defects or hard faults – these become simply part of the (desirable) randomness in the structure of the computational core; and (3) its natural robustness to structural noise caused by performance variability/fluctuations, which, as will be seen, can be effectively 'filtered out' during the task-dependent machine configuration phase. These three points make perturbation-based computing very promising for technologies exhibiting the high defect densities and substantial performance and structural uncertainty projected for emerging nanoelectronics. At the same time, characteristics of perturbation-based computing that make it promising to address the challenges and needs of next generation IT systems, include: (4) its suitability for implementing the many soft real-time stream processing and reactive control tasks that will comprise such systems; and (5) the limited design effort required, in that, as we will show, this computational model can be realized/implemented on configurable platforms, usable for many different tasks.

To establish the promise and potential of perturbation-

This work is supported by the Gigascale Systems Research Center under the 'Alternative' Theme.
This paper is an extended version of work presented at Nano-Net 2008.

based computing, this paper proposes a novel hybrid eNano-CMOS platform for realizing such machines – as will be seen, the platform relies on a new style of configuration that, we believe, can directly leverage the strengths and circumvent the limitations of technologies characterized by high density but also high structural and performance uncertainty. We further identify and demonstrate a new set of fundamental *design principles* and *decomposition strategies* which are effective for perturbation-based computing platforms, and propose a multi-core machine architecture which exposes these principles. The importance and impact of this second set of contributions lies in establishing that this new class of computational models will scale and is amenable to systematic design, two key practicality concerns. These points are empirically demonstrated for a representative set of soft real-time processing tasks from a variety of domains.

To perform these experiments, we substantially enhanced a publicly available simulation tool [12], so that it could support the large variety of machine configurations relevant to our study, including: (1) distinct computational core realizations (e.g., using different processing nodes and/or connectivity constraints); (2) operation in discrete and continuous time; and (3) multi core machines organizations, exposing multiple/differentiated core dynamics – see Sections II and IV for details.

Note that the class of computational models investigated in this paper was recently discovered, independently, by two research groups [12], [13]. Yet, their work was driven by research pursuits and objectives quite different from those in this paper. Section V gives details on such prior work and establishes the uniqueness and novelty of our paper's contributions.

Organization. The remainder of this paper is organized as follows. Section II provides background on perturbation-based computational models. Section III introduces our proposed hybrid eNano-CMOS configurable platform and makes the case for its use for next generation IT. Section IV presents a novel set of design and decomposition principles for perturbation-based computing, a machine architecture exposing such principles, and demonstrates their effectiveness using concrete experimental data. Section V contrasts the work presented in this paper to relevant previous contributions, and Section VI concludes with a discussion on future work and challenges.

II. Background: The Principles of Perturbation-Based Computing

Key idea. Perturbation-based computational models are ideal for implementing complex non-linear filters (operators) associated with real-time information processing. The key idea is to perform a non-linear projection of the input stream into a high dimensional space using a complex dynamical system. If the pool of dynamics capturing information about current and past stimuli is sufficiently rich, *any* desired non-linear filtering task's output(s) can be derived, or 'composed' from it. Below we develop this basic idea in a more rigorous manner.

Mathematical foundations. The fundamentals of perturbation-based computational models can be traced back to a result of Boyd and Chua on approximating time invariant nonlinear operators that have fading memory[14].

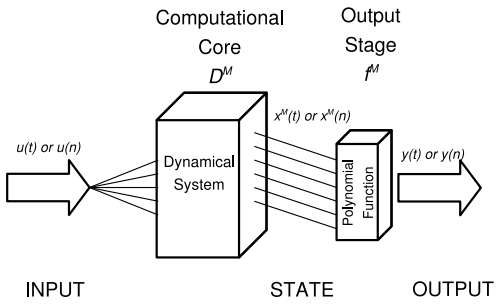


Fig. 1. A Perturbation-based machine.

Namely, they showed that such operators on bounded Lipschitz continuous (i.e., slew limited) inputs can be approximated arbitrarily closely by a *finite* Volterra series operator. As informally stated by Volterra, see [14], the fading memory[1] requirement means that "the influence of the input a long time before the given moment gradually fades out."

Maass [12], one of the original proponents of this computational model, essentially re-states the above result for multidimensional inputs, as follows: a continuous, multidimensional time invariant operator $F : \mathbb{R}^{\mathbb{R}^n} \rightarrow \mathbb{R}^{\mathbb{R}^k}$ with fading memory can be approximated arbitrarily closely by an operator $F^m : \mathbb{R}^{\mathbb{R}^n} \rightarrow \mathbb{R}^{\mathbb{R}^k}$ consisting of two elements [12]. The first is a *finite* set of basis operators $D^m =< O_1, \cdots, O_M >$ where $O_i : \mathbb{R}^{\mathbb{R}^n} \rightarrow \mathbb{R}^{\mathbb{R}}$ are selected from any family $\mathcal{O}$ of operators with fading memory satisfying the *pointwise separation property*. This requires that $\mathcal{O}$ have sufficient 'discriminating power' – specifically, given any two distinct inputs u, v, there exists an operator $O \in \mathcal{G}$ such that $Ou \neq Ov$. There are many families of operators satisfying this property, including: the class of all delay operators U_τ where $U_\tau u(t) = u(t - \tau)$; the class of all linear operators with exponential impulse responses $h(t) = e^{-at}$, with $a > 0$; and the class of non-linear operators defined by standard models for dynamic synapses [12]. Given an input $u \in \mathbb{R}^{\mathbb{R}^n}$, we let $x^m(t) \in \mathbb{R}^m$ denote the vector output of these operators at time t, i.e.,

$$x^m(t) = (D^m u)(t) =< (O_1 u)(t), ..., (O_m u)(t) > .$$

The second element is a *memoryless* polynomial readout function f^m, or approximation thereof. The output at time t is denoted by $y(t) \in \mathbb{R}^{\mathbb{R}^n}$ and given by a composition of the set of operators and the memoryless function :

$$y(t) = f^m(x^m(t)) = f^m((D^m u)(t)).$$

Additionally, one can show that the discrete time counterpart of this problem is fairly simple, in that the approximation can be realized by a simple nonlinear moving average operator [14].

Perturbation-based machines. Fig. 1 symbolically depicts a perturbation-based machine M. As can be seen, it maps

[1]Formally, an operator with fading memory satisfies a slight strengthening of the natural continuity condition. Specifically, an operator is said to be continuous if input signals that are close (i.e., have a small peak deviation over all past time) have present outputs which are also close. However, in the case of an operator with fading memory, it suffices for the inputs to only be close in the recent past for the outputs to be close [14].

an input function $u(\cdot)$ to an output function $y(\cdot)$, relying on two key components: a high dimensional dynamical system, implementing the machine's *computational core* D^M, and an *output stage* f^M. The key premise underlying perturbation-based computing is that, by using computational cores realized by sufficiently complex, even random, dynamical systems, one can essentially project inputs over a sufficiently large family of basis operators for any given set applications and desired approximation level [12]. A machine's D^M is thus a dynamical system realizing a very large pool of candidate operators, while the abovementioned D^m denotes a specific set of basis operators required for a given approximation. As such, the *same* computational core D^M can be used in realizing various tasks. The output stage is the *task dependent* part of this machine, playing the role of both selecting and composing the 'relevant' basis operators through a memoryless function.

As shown in Fig. 1, the computational core D^M generates an internal state $x^M(t)$, corresponding to a causal response to the input u. This is a non-linear projection of the input stream on a high dimensional space, generated by exciting the dynamical system associated with D^M. Note that *no stable internal states* are required in the computational core, it suffices to generate a sufficiently rich pool of transient dynamics. As such, one can say this computational model is *non-Turing* – a key departure from conventional computational models. The output stage f^M maps the internal transient state to a specific output.

Performance limits and approximation. Based on Boyd and Chua's fundamental result, Maass established that perturbation-based machines have universal computing power – that is, machines operating 'natively' under this computational model can approximate *arbitrarily closely* any time invariant fading memory operator [12]. Still, although Boyd's result tells us that the number of basis operators required by any such approximation is *finite*, it says nothing about how many such operators may be required in each case. If very high precision is required, the number of operators may be relatively high for certain tasks. It is however important to note that such cost/accuracy tradeoffs may be of interest in certain applications. Indeed the proposed computational model is inherently based on realizing approximations, so, with the exception of very simple functions/operators, it is not expected that to operate without error. Thus we target applications where this is unacceptable, and in fact presents an opportunity to tradeoff error rate against other costs, e.g., manufacturing cost, power consumption etc. An example of such a task would be real-time searches for block matching across video frames, a task which is essential in video compression. When the best match is missed, the algorithm does not fail, instead a temporarily lower compression rate results. Another class of applications involves systems with feedback, where occasional errors can be subsequently compensated via feedback resulting in an overall negligible effect. Real-time multimedia processing and control applications will be a pervasive and important subset of the emerging classes embedded information processing infrastructure.

Modeling perturbation-based machines. In practice, the dynamical system comprising the machine's computational core, D^M, can, for example, be realized by a complex (randomly generated) recurrent network of non-linear operator nodes [12], [13]. In fact, given the rich pool of dynamics generated by such networks, the machine's task dependent output function can, in general, be quite simple, e.g., *linear*. Accordingly, for all experiments reported in this paper, only linear readout maps were considered. As such, we have used standard linear techniques to determine appropriate output functions: linear regression for tasks with real-valued outputs, and linear classification for discrete outputs [15].

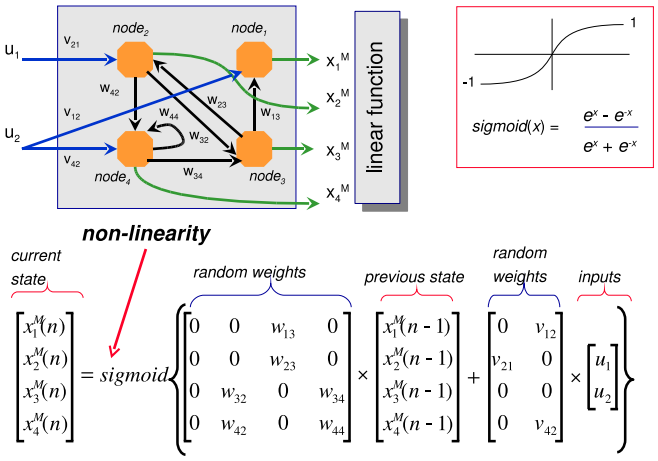


Fig. 2. Computational core and output stage of a small discrete-time perturbation-based machine.

Relying on the formal definition and broad principles given above, one can still build many variants of a perturbation-based machine, e.g., operating in discrete or continuous time, relying on different types of non-linear nodes, etc. For illustrative purposes, Fig. 2 shows an instance of a perturbation-based machine operating in discrete time, where each of the core's nodes apply a sigmoid function (scaled to the range $[-1, 1]$) to a weighted sum of their inputs. Due to space limitations, we depict a very simple computational core comprised of a recurrent network with only 4 nodes. (For the actual experiments reported later in the paper, much larger sparse incidence matrices defining the connections and weights of the corresponding recurrent networks were randomly generated.)[2] As shown in Fig. 2, the next state of the computational core (i.e., of each of its nodes $x_i^M(n)$) is computed based on the core's previous state and the current inputs. In turn, the task dependent linear readout function at the machine's output stage is defined by assigning a corresponding weight (in the picture, denoted k_i for node n_i) to each of the core nodes.

Representative perturbation-based machine. Most experimental results presented in this paper were generated by simulating machines operating under the discrete time model illustrated in Fig. 2. This model is very similar to the ECHO model proposed in [13], except that in our case: (1) node-to-

[2]Note that in order to avoid chaotic behavior, such randomly generated sparse matrices were then scaled, so that the absolute magnitude of the maximum eigenvalue is 0.95, see details in [15].

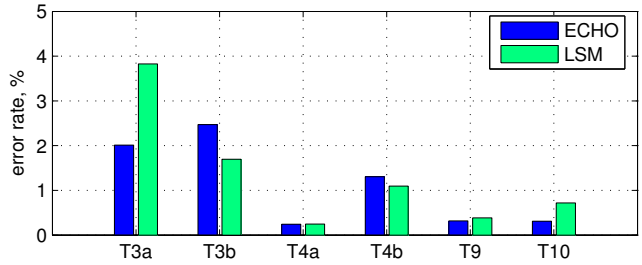


Fig. 3. Average task error rates for a pool of machines with randomly generated computational cores, operating under ECHO-like and LSM models.

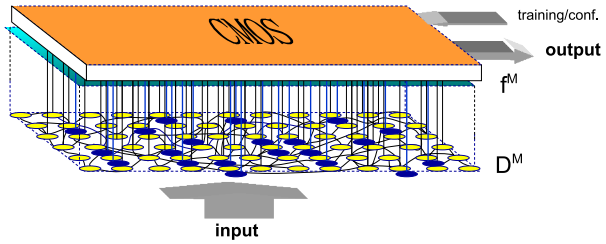


Fig. 4. Hybrid eNano-CMOS configurable platform for perturbation-based computing.

node connections within the computational core where generated introducing a strong bias towards local connections, so as to reflect practicality concerns[3]; and (2) there is no feedback loop projecting the output back to the computational core, as in the standard ECHO model. This machine is somewhat abstract, since the sigmoid non-linearities would be complex to implement in practice. We also experimented with another radically different network and node model, based on leaky-integrate-and-fire nodes operating in continuous time, as in [12] – we refer to this as the LSM model. The performance results in Fig. 3 exhibit the error rates for the LSM and ECHO-like models, across several tasks. (The set of benchmark tasks and experimental methodology are detailed in the Appendix.) As can be seen, we systematically obtained equivalently good results, supporting our claim that the specifics of the internal network dynamics are not of great importance, as long as they are sufficiently rich. Since simulation of discrete machines with sigmoid nodes is much faster, subsequent results in this paper will be based on our ECHO-like model [13], which we deem to be broadly representative.

III. A Hybrid eNano-CMOS Configurable Platform for Perturbation Based Computing

Proposed configuration paradigm. Perturbation-based computational models enable a new configuration paradigm that is uniquely suited for technologies characterized by high densities and high manufacturing and performance uncertainty. As mentioned earlier, our intent is not to design dynamical systems to realize specific base operators for a given task. Instead, we propose to embrace the inherent uncertainty intrinsic to nanoscale technologies, and generate a random pool of *candidate* basis operators which is sufficiently rich to approximate the task at hand. The proposed configuration paradigm is thus as follows: design machines with dynamical systems which provide a large pool of possible basis operators, and then select/discover the subset needed to approximate the task of interest. Extensive empirical data generated by Maass, Jaeger, and others, including ourselves, shows that *randomly generated* complex recurrent networks provide sufficiently rich pools of base operators – see experimental results below. Note that in the aforementioned experiments, connections and weights amongst nodes in recurrent networks were randomly generated, yet the effect of structural uncertainty can be further incorporated into heterogeneity in the non-linearities associated with nodes themselves.

From function to functional approximations. Typical approaches to function/classifier approximation are based on selecting a good approximation from a parameterized set of functions. In general selecting the best approximation may involve solving an optimization that is not necessarily convex, i.e., may have local minima. A typical example is multi-level perceptron. When gradient descent is used to train it, one can end up in some local minimum. A special case is that where approximations are based on weighted combinations of a set of finite set of possibly nonlinear basis functions. With typical approximation costs, and a linear dependence the weights a unique solution can be determined via gradient descent with a hardware friendly implementation. In principle one could consider taking linear combinations of *random* basis functions. In this case one can still argue training would converge to a global minimum. Though we can not expect very high approximation accuracy if complex functions are being approximated, or the sample space of basis functions is not sufficiently rich. The usefulness of this approach lies elsewhere; in its simplicity, generality, and the potential to make it hardware friendly. When faced with the task of approximating or learning dynamics, i.e., functionals or operators, one can use a similar approach. The random basis functions are now replaced by random basis operators that are used to approximate desired operator using linear combinations. One can not expect very high approximation accuracy of complex operators at small cost. Yet the basic approach can be used as general building block to implement such operators using higher level techniques (e.g. using hierarchical task structure or some other structure). The realization of such functional approximations based on randomly assembled networks, i.e., sets of basis functions, is the key idea in this paper.

Hybrid platform. Given the previous configuration paradigm, we propose the use of a hybrid eNano-CMOS platform for perturbation-based machines, where the machine's computational core is implemented on an emerging nanoelectronic fabric while CMOS is used to implement the simple (e.g., linear) read out function at the output stage and support the machine configuration/training process. Fig. 4 shows an abstract view of such a platform, with the key basis operators in the pool highlighted in bold. Clearly,

[3]Concretely, when generating a machine core, we embed its corresponding nodes on the integer points of a 2D or 3D grid. Then, as done in [Maass], relying on the resulting Euclidian distances between core nodes, we randomly choose connections between them, using a probability low favoring shorter/local connections.

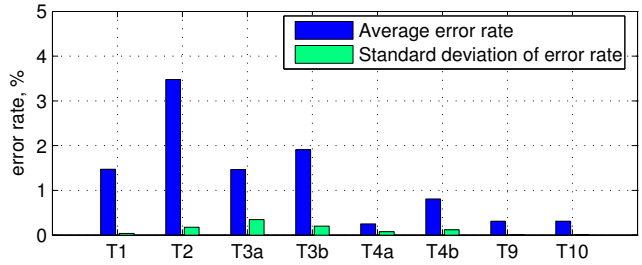


Fig. 5. Average and standard deviation of task error rates obtained for a pool of machines with randomly generated computational cores of a target size.

this platform can directly leverage the formidable densities achieved by nanotechnologies to create computational cores of essentially arbitrary size. Furthermore, since the recurrent networks used to implement such dynamic systems would in principle be 'randomly' assembled, the need to design and precisely manufacture structured circuits is to a large extent circumvented. Fig. 5 empirically supports this argument. It shows that, given a benchmark task, machines with randomly generated computational cores of a similar (sufficiently large[4]) size exhibit negligible variation in their ability to perform the task, i.e., have essentially the same computational power – assessed based on task error rates. Details on the experiment's eight benchmark tasks and experimental methodology are given in the Appendix.

The proposed approach requires one to perform a training step for each chip. This is indeed a costly requirement. Yet these overheads might be 'similar' to those associated with typical defect tolerance approaches. Indeed the typical requirements in the latter are to detect, i.e., map out, defects for each chip and then resynthesize the function to avoid defects. Defect mapping is typically done using test patterns that are either obtained/generated off chip or stored on chip. Resynthesis involves reprogramming the function around the defects on the chip. In our case rather than defect mapping and resynthesis steps we require a training step. Such training will involve access to input-output pairs that can also be provided either off-chip or on-chip. A comparison of the cost of mapping an resynthesis vs training is at this point premature.

Robustness to structural uncertainty: defects and performance variability. Another important advantage of this computational model is that defective nodes in the computational core are naturally circumvented when the relevant basis operators are selected during the configuration process. Accordingly, the high density of hard defects projected for nanotechnologies would simply become an integral part of the structural heterogeneity of the recurrent network(s) implementing the cores, posing no harm to the eventual performance of the machine.

As will be shown, *dynamic* performance variability, which is intrinsic to nanoscale regimes, is also naturally tolerated by perturbation-based machines. Such variability is likely to be observed in the operation of, both, nodes and interconnects and should can be viewed as additional 'run-time' *structural*

[4]See the Appendix for a discussion on the core sizes selected for this experiment.

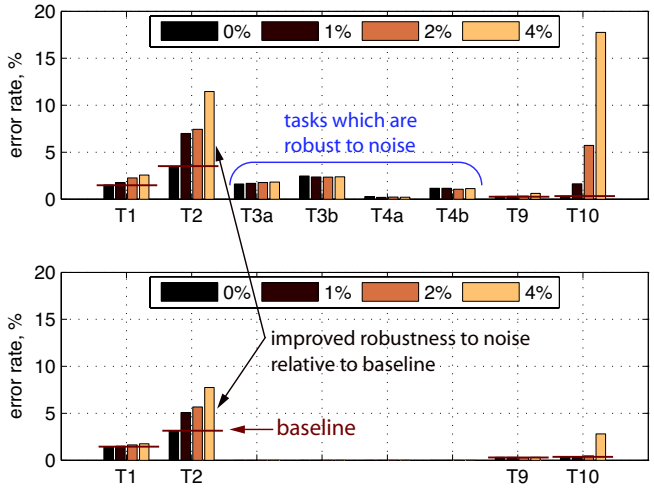


Fig. 6. Average task error rates for a pool of machines with randomly generated computational cores, operating under three noise regimes.

noise impacting the 'nominal' response of each element defined upon fabrication.[5] Accordingly, we assessed machine performance (task error rates) in the presence of structural noise resulting from dynamic performance variability in the computational core. Fig. 6 exhibits results for the following three noise regimes (assuming identical core sizes in all cases):

Scenario 1: no noise, serves as our baseline.
Scenario 2: zero mean additive (uniformly distributed) white noise affecting *all* nodes of the machine's processing core, intended to model signal perturbations resulting from device and interconnect variability. Level of noise is within 1% of the actual signal range.
Scenario 3: same as above, with noise ranging within 2% of the actual signal range.
Scenario 4: same as above, with noise ranging within 4% of the actual signal range.

As can be seen on the top in Fig. 6, even for the highest noise level, the error rate increases are in most cases fairly small for identical core sizes, supporting our claim that this computational model operates well under this type of persistent performance variability. Furthermore, on the bottom in Fig. 6 we show how the error rate increases for tasks which were not robust to noise can be dramatically reduced by simply increasing core size and density of connectivity – in this case by a factor of two and three, respectively. Thus, the design of cores can mainly rely on proper broad sizing of the core's network to achieve the desired reliability under performance uncertainty – we refer to this as 'unstructured redundancy.' This should be contrasted with the design complexity associated with the structured redundancy required by machines operating under traditional computational models.

[5]Indeed, at such reduced scales, the discrete nature of atomic matter and charge becomes significant, and nanodevices and interconnects will exhibit great sensitivity to fluctuations in the local electrostatic environment, e.g., even a single charge may significantly impact a nanodevice's timing/performance.

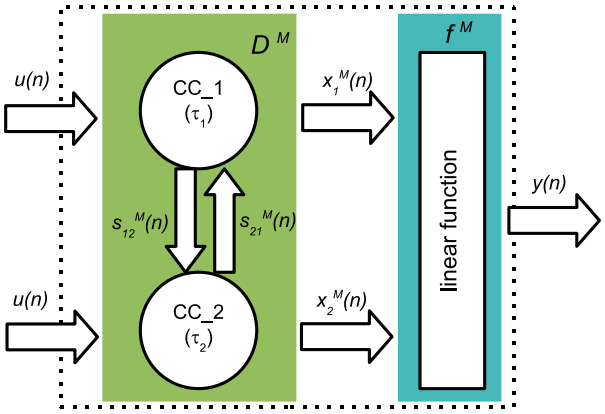


Fig. 7. Multi-core machine architecture.

IV. A Machine Architecture for Perturbation-Based Computing

So far our discussion of perturbation-based machines assumes that they would contain a single monolithic computational core, yet in what follows we show that the overall flexibility and scalability of this computational model can be greatly enhanced by considering machine architectures incorporating a multi-core organization, see e.g., Fig. 7.

Proposed multi-core machine architecture. A shown in Fig. 7, we envisage an architecture that has multiple computational cores (i.e., reservoirs of dynamics) of one or two basic standard sizes, and where the dynamics' speed of the cores can be tailored to specific classes of applications. Accordingly, each core has a rate parameter, τ_i, representing the inherent speed of its dynamics, e.g. for a discrete system it might simply correspond to slow updates, whereas for a continuous system corresponds to the dynamics' relaxation time. The importance of this parameter will become clear in the sequel. Additionally, as shown in the figure, each core can be individually excited by a subset, or all, of the inputs, and may access state information from other (neighboring) cores, through a strong or weak coupling. Finally, the task's readout function in the output stage may rely on state from all or just a subset of the computational cores. In the sequel, we illustrate a number of different machine configurations along with basic decomposition principles.

Combined core size and computational power. Consider first a perturbation-based machine comprised of a single monolithic computational core. As one would expect, the computational power of such a machine can be enhanced by increasing the size of its computational core, thus creating a richer pool of operators. Experimental results for monolithic machines shown in Fig. 8 (denoted by an *m* in the figure's legend) illustrate this – as can be seen, machine performance for a set of benchmark tasks improves with the size of the underlying computational core, until it nearly saturates, once a sufficiently rich pool of dynamics is generated.[6] In addition, Fig. 8 shows the same set of results, but now generated using a multicore machine (denoted by *c* in the legend) with a combined core size identical to that of the original monolithic machine – in this experiment, each individual core has 100 nodes, and thus a 200 node machine uses two cores, a 400 node machine uses four, and so forth. Furthermore, a decoupled core configuration was adopted, i.e., the internal state of a core is not accessed by any other core. As can be seen in Fig. 8, the performance of the multicore machines is essentially the same of the monolithic core machines, suggesting that one may create flexible platforms with many small cores, and then use/configure only those necessary to achieve the required computational power for the task(s) at hand.

[6]After this saturation point, increasing accuracy relying strictly on the random nature of the computational core, becomes increasingly expensive.

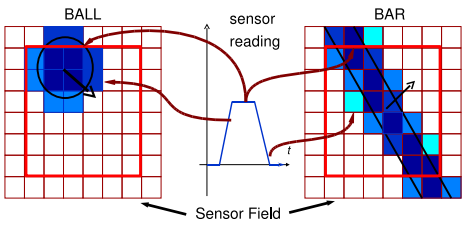


Fig. 9. Vision task: predicting object movement on an $8x8$ array of sensors.

Note further that, depending on the task, simply increasing the combined core size, even before the saturation point alluded to above is reached, may be a poor design strategy, and may also unnecessarily bound the practically attainable task accuracy. That is, task performance may be more effectively enhanced by imposing a more suitable alternative decomposition on the multiple computational cores.

Core decomposition driven by inherent structural/physical locality characteristics of a task. For some real-time processing tasks, there is a natural partition of inputs into subgroups which are known *a priori* to capture features whose *dynamics* need not have a strong interaction in performing the resulting task. Observe that this does not mean that the output does not depend jointly on all input subgroups, but rather that relations between the intrinsic dynamics of the input subgroups are essentially 'independent' features on which the readout function should draw. When this is the case, rather than jointly project all input streams into a common set of shared cores, it is more computationally effective to feed these inputs into separate subsets of cores which are only weakly connected, if at all.

Task 8 is used to illustrate the effectiveness of this decomposition principle, and how it can be applied in practice. This is a motion prediction benchmark – a moving object crosses an 8×8 field/array of sensors, with a random but constant speed and direction – see Fig. 9, where we use color intensity to represent the percentage of each sensor field currently covered by the moving object [16]. With equal probability, the object is a ball or a bar. The task is to predict the readings of the inner 6×6 array of sensors, one step in advance. We performed two experiments for this task, one where the machine has a single monolithic core and a second where it has multiple cores. The monolithic core has $16 \times 16 \times 3$ nodes (totaling 768 nodes) [16]. In turn, reflecting the structural characteristics of the sensor field in Task 8, the multi core machine comprises one core per sensor – thus a total of 64 cores with $2 \times 2 \times 3$

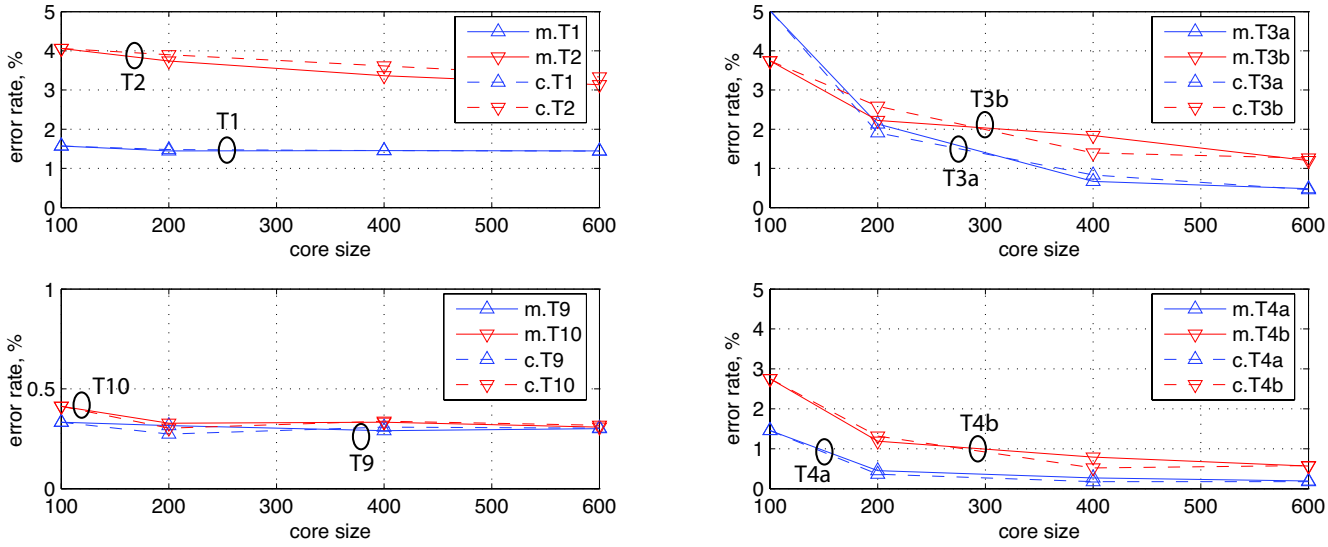


Fig. 8. Test error rates for benchmark tasks, considering monolithic core machines ('m') with increasing core sizes, and multi-core machines ('c') with the same combined core sizes.

nodes each – giving the same total of 768 nodes. Each small core is weakly connected to its four nearest neighbors in the grid, receives only its own local input, i.e., the corresponding sensor reading, and outputs the next step reading prediction for that sensor. For simplicity, our predictions consider only two possible values: ON, corresponding to most of the sensor field (i.e., $> 60\%$) covered by the object at the next time step; or otherwise OFF. All cores operate at the same speed.

For this experiment, the monolithic core machine gave test error rates of 2.55% for ON predictions and 0.10% for OFF predictions, while the multi core machine gave test error rates of 2.63% for ON predictions and 0.10% for OFF predictions, i.e., delivered similar performance to that of the machine with the large monolithic core. The advantages of this arrangement, from a scalability and efficiency standpoint, should be clear, in particular if one considers larger fields of sensors.

Core decomposition driven by task dynamics which are 'nearly decomposable'. It is not unusual for real-time dynamical systems to exhibit a 'nearly decomposable' structure [17]. For example, the input streams might be divided into subgroups associated with characteristics that are varying on different timescales. In this case, rather than directly mixing input streams with different timescales in shared cores, it may be more effective to structure the computational resources to reflect the task at hand. For example, consider a system with two intrinsic dynamical time scales, a fast and a slow one. If, from the perspective of a task, these characteristics were independent, i.e., decomposable, then one might feed associated inputs into independent sets of cores and allow the readout function to draw from the two types of reservoirs. Alternatively, one might have fast dynamics which are conditioned on slow dynamics in a system. In other words, the slowly varying input characteristics set the broader 'context' for fast varying characteristics of the system. If such a time scale decomposition is not performed, one may (unnecessarily) require a very large pool of dynamics, operating at the fastest speed, in order to achieve good performance, whereas a set of properly interconnected small cores operating at multiple time scales could provide a very effective set of projections for the task at hand.

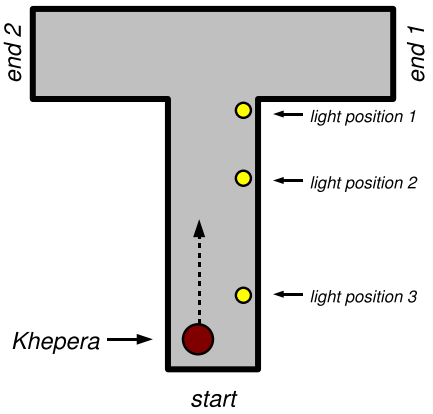


Fig. 10. Three robot navigation tasks on T-Maze: Task 5 (light at position 1), Task 6 (light at position 2), and Task 7 (light at position 3).

Below, we present results for an experiment illustrating the scalability enhancements one can achieve via this unique type of decomposition. We consider the classical T-maze robot navigation task – as shown in Fig. 10, the task is to have a robot (Khepera [18]) navigate to *end 1*, if light is on, otherwise go to *end 2*. The initial robot position (in the start region) is randomly selected. The task inputs and outputs are the robot's 16 sensor readings (8 light and 8 proximity sensors), and the speeds of its two wheels, respectively. Three benchmark tasks – 5, 6 and 7 – have been defined, each corresponding to a different position of the light. Namely, as shown in Fig. 10, for Tasks 7 and 6 the light is at the very beginning and at some

intermediate point of the corridor, respectively, while for Task 5 the light is at the T-junction, i.e., right at the decision point.

Before presenting the results of this key decomposition experiment, it is important to introduce an additional notion critical to this work – *yield* – which is defined for a given machine configuration, target task, and desired performance. Recall that, in Fig. 5, for example, we considered eight benchmark tasks, and presented the corresponding *average* and *standard deviation* of their error rates, derived for a pool of machines with randomly generated computational cores of a given target size. As indicated, for that experiment the target core sizes were chosen to be *large enough* so as to *saturate* performance. That is, any core size increases beyond their associated values would not enhance task performance in any substantial way, thus ensuring low variability on task performance across our randomly generated machine pools (more details are given in the Appendix).

Yet, rather than design up front such a large/costly machine, one may choose instead to use smaller computational cores and accept a lower yield for the task of interest – that is, given the target task and desired performance, one may be willing to discard a certain percentage of machines (in the random pool) that are unable to deliver the required performance. It should be clear from our previous discussions that by limiting the size of the computational core, certain machines within a random pool may not be configurable to deliver the desired performance. Still, the relevance/attractiveness of explicitly considering such yield-related tradeoffs is the possibility of using smaller and least costly machines to execute the tasks of interest, while meeting the required performance. Thus, by incorporating this additional *design space exploration dimension* – yield – we can provide more insightful experimental results for this class of machines.

Accordingly, given a machine configuration, target task, and desired performance, in the sequel we refer to the percentage of machines that can deliver the desired task performance (when a large pool of such machines is randomly generated) as yield. Let us now return to the T-maze experiment. We present results considering two performance targets (less than 2% error rate (ER), and less than 10% ER), and two yield points (75% and 50% of the machines in the random pool generated for the selected configuration are able to deliver the required performance or ER).

We first assessed the performance of *monolithic core* perturbation-based machines, for the three T-maze tasks. For Task 5, we found that machines with a relatively small computational core (comprising 75 nodes) were able to deliver the top performance target ('less than 2% error rate'), with a yield of 75% (the maximum yield point considered in this experiment). Furthermore, we found that a smaller machine configuration (comprising only a 50 node core), was still able to deliver our top performance ('less than 2% error rate'), yet with a decreased yield of 50% (our second/lower yield point). Thus, for this smaller configuration, only half of the machines (rather than two thirds) would operate within the desired target error rate of 2%. Our experiments also showed that this same small configuration (with a single 50 node core), was able to deliver our lower performance point ('less than 10% error rate'), with an improved target yield of 75%. This set of experimental results clearly illustrates the richness of the yield-related tradeoffs introduced above.

We now present and discuss the results obtained for Task 6 – with the light placed at some intermediate point of the corridor (see Fig. 10). In contrast to Task 5, much larger core sizes were consistently required to achieve similar performance and yield targets. For example, to deliver our lower target performance ('less than 10% error rate') with our max yield point of 75%, a machine configuration comprising a single core with 300 nodes was required by Task 6 – this should be contrasted with the 50 node core required by Task 5. In turn, to deliver our top target performance ('less than 2% error rate') with the low target yield of 50%, a machine configuration comprising a single core with 600 nodes was required by Task 6 – in contrast to the much smaller Task 5's 50 node core. Note further that we are unable to give concrete numbers for Task 6's top performance and yield points (i.e., 'less than 2% error rate' with a 75% yield), since we performed experiments only with core sizes of up to 1000 nodes, and those targets were not met even by machines with such large core sizes – while Task 5 required cores with only 75 nodes.

Finally, for Task 7, with the light placed at the very beginning of the corridor (see Fig. 10), we were not able to achieve any of our selected performance vs. yield points, even for a machine with 1000 nodes. The problem is that, as the distance from the light to the T juncture increases, even a large machine with a single 1000 node computational core is not able to 'remember' the state of the light when the robot reaches the decision point. As empirically demonstrated by the experiments shown in the beginning of this section, a multi-core machine with merely a combined core size identical to the maximum monolithic one indicated above (i.e., 1000) would deliver a similar performance, and thus not address the very poor performance observed for Task 7.

To address this issue, for Tasks 6 and 7 we considered alternative machine configurations with two processing cores, each with a different τ parameter. One of the cores is responsible for the 'larger scale' context (or slower dynamics) associated with the task, i.e., 'remembering' the light and what to do in both cases, while the other core takes care of immediate (or fast dynamics) navigation decisions, i.e., staying away from the walls of the corridor while moving forward. The slow core is excited by the robot's light sensors only, whereas the fast core is excited by the light and the proximity sensors as well as the slow core. The task's readout function, which is responsible for generating the speed of the two independent robot wheels, relies only on state from the fast core.

Fig. 11 summarizes the results delivered by such decomposition for Task 7 (the 'harder' task). The top graph of Fig. 11 gives the combined core size of several machine configurations capable of delivering our top performance point (error rate under 2%) with a yield of 50% and 75%. Namely, for each slowdown factor in a range from 30 to 100 (corresponding to a different τ parameter), the graph gives the size of the smallest machine configuration that can deliver an error rate under 2% with the particular yield. In turn, the bottom graph shows those same combined sizes, but now for our lower target

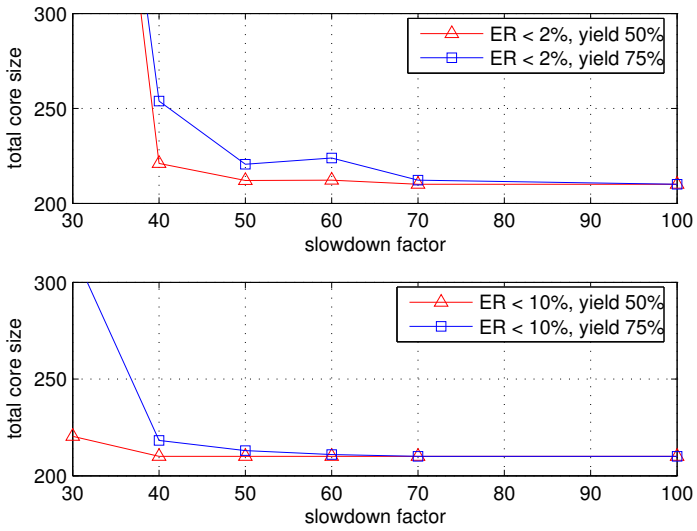


Fig. 11. Size of double core machines for T-maze task T7, considering different relative core speeds, and different target performances and yields. On all experiments, the size of the fast core is 200 nodes. The top and bottom figures show the combined core sizes required by machine configurations exhibiting a target error rate (ER) of less than 2% and 10%, respectively, as the slowdown factor of the slower core increases.

performance point (error rate under 10%). The enhancement in efficiency and compute power achieved by such dual-core dual-dynamics machines is remarkable – for example, as can be seen in the top graph, a machine provided with a fast and a 40 times slower cores with only 200 and 50 nodes, respectively, can achieve the target 2% error rate for Task 7 (the 'harder' task) with a yield of 75%. Recall that, by contrast, the error rates delivered by a 1000 node single core machines for this task were extremely high. This represents a dramatic improvement in task performance and reduction core size empirically supporting the effectiveness of the proposed decomposition.

The results in Fig. 11 give us other interesting insights. Namely, as we can see in the top graph of the figure, as we move below 40, towards 'small' slow down factors, the required size for the slow computational core increases, as one would actually expect – ultimately, as the slow down factor reaches 1 (denoting both cores operating at the same speed), we would need a combined size essentially identical to that of the monolithic core configuration previously discussed. In contrast, as one moves right, towards 'large' slow down factors, the size of the core decreases, until it stabilizes into one that is large/rich enough for the dynamics that need to be captured. Beyond those basic trends, making more precise assessments on what would be the best size vs. slowdown factor (e.g., in terms of resulting core activity, cost, etc.) would be more sensitive to the type of perturbation-based machine being considered. Since our experiments rely on a representative, yet very abstract, machine, it suffices for now to show the basic trend inherent to the decomposition.

Finally, as one would expect, the dual-core dual-dynamics machine delivers substantial scalability gains for the less challenging Task 6. For example, we found that a dual-core dual-dynamics machine with relative core speeds differing by a factor of 20, requires a combined core size of only 150 nodes to deliver our top performance target (2% error rate), with the max yield point (75%). By contrast, as mentioned above, a monolithic core machine requires 300 nodes to achieve the *low* performance target of 10% error rate with the same yield (75%).

These experiments empirically demonstrate the enhanced flexibility/practicality as well as substantial scalability gains possible with our proposed multi-core machine architecture, which: (1) relies on a set of standard size cores, either only locally coupled or fully decoupled; (2) allows for the joint projection of select subgroups of inputs, rather than always requiring the joint (brute-force) projection all inputs; (3) allows the readout function to rely on the state of only a select subset of cores; and, last but not least, (4) enables exploitation of multiple speed dynamics during machine operation, by allowing cores operating at different time scales τ. These novel decomposition principles have empirically shown to be strikingly effective in the context of perturbation-based computing.

V. Contrast to previous work

Contrast to other computational models. As mentioned above, perturbation-based computing relies on transient internal states, and is therefore not a Turing model. That is, unlike Turing machines, or even less powerful standard finite state machines, the only stable state of a perturbation-based machine's computational core is in general the 'rest' state, and the machine has no 'permanent' memory. These characteristics make this computational model unique. Furthermore, although perturbation-based computational models are likely to use recurrent networks to realize their computational core, they are also fundamentally different from traditional recurrent neural networks (RNNs). Indeed, after training, RNNs operate essentially as deterministic FSMs. That is, they exhibit a finite number of attractor states, which encode their possible outputs. A key challenge for RNNs is designing system dynamics so as to create suitable stable/attractor (or 'low energy') states, so that a network subject to specific inputs eventually converges to the correct stable states. Such convergence is hard to achieve, since dynamic systems may easily become unstable – thus, most research in the field deals with very simple special cases or network topologies [15]. By contrast, the operation of perturbation-based machines does not center on designing/controlling dynamics, but rather draws on the rich transient dynamics of complex (and possibly randomly assembled) systems, operating under substantial structural uncertainty.

Previous research on perturbation-based computing. As mentioned earlier, perturbation-based computing was independently proposed by two research groups [12], [13], both of which have furthered this area, but have fundamentally different objectives than ours. The main objective in [12] was to model *biological* neural circuits and understand the operating principles of such biological systems. Accordingly, ensuring biological plausibility (e.g., for the network nodes), and successfully mimicking the behavior of actual neocortex

circuits (e.g., in terms of possible information encoding), were key drivers for their research [12], [19], [16]. These issues are not germane to our research, and in fact obscure our core objectives. The work of [13] was driven by the desire to develop practical engineering techniques for training (artificial) recurrent neural networks, to be used in control applications. Their main objective was to circumvent the need to control (design) complex network dynamics – an exceedingly hard problem [15]. Note, however, that this line of work assumes that such networks/models are to be directly programmed, using Turing complete languages, on conventional general purpose computers. By contrast, our focus and contributions are directed towards realizing machines that operate directly under this computational model, rather than emulating or simulating it using Turing complete machines/languages. In particular, we are interested in assessing the suitability of perturbation-based computational models for nanotechnologies characterized by high defect density and high performance variability – a major research challenge posed to the computer science and computer engineering communities.

VI. Conclusions and Future Work

We proposed a hybrid eNano-CMOS platform for realizing perturbation-based machines, relying on a new style of configuration that, we believe, can potentially leverage the strengths of emerging nanoelectronic technologies. In particular, we presented experimental evidence demonstrating that hard defects and performance variability/uncertainty can be naturally handled within this framework. To assess the scalability and practicality of perturbation-based computational models, as well as develop the foundation of tools for engineering efficient systems relying on these, we then proposed a multicore machine architecture and novel design and decomposition principles, exploiting task specific contextual and temporal scales. We experimentally demonstrated the effectiveness and promise of these, for a set of benchmark tasks.

Given the promising results reported in this paper, a key objective for future work is to devise simple node nonlinearities – ideally exhibiting a complexity of one or two gates equivalents – such that a single computational core with 50, 100 or even 200 nodes would be still quite a small component for today's standards. Beyond simplicity, additional critical requirements are that such nodes should be possible to successfully assemble/fabricate under substantial structural uncertainty, i.e., be robust to spatial variability during assembly/fabrication. These implementation oriented, research topics make sense in the context of a concrete architectural framework and corresponding promising results, as presented in this paper. Last but not least, it is paramount that, during assembly, core nodes form recurrent networks exhibiting non-chaotic behavior (e.g., by exhibiting appropriate dumping factors), either by construction or through some simple post-fabrication process. In [20], some promising adaptive directions are explored towards this end, for a class of abstract perturbation-based machines which we denote standard ECHO model [13]. Still, in our case it will be paramount to incorporate challenges specific to nanotechnologies into the process, as well as operate in the context of a more concrete architectural framework, such as the one presented in Fig. 4.

We conclude by observing that it will be also interesting to consider heterogeneous systems combining perturbation-based computing with more traditional computational models. Yet, this direction makes sense only after the fundamentals and strengths and limitations of perturbation-based computing are better understood and analyzed.

References

[1] V. Zhirnov and D. Herr, "New frontiers: Self-assembly and nanoelectronics," *IEEE Computer*, vol. 34, no. 1, pp. 34–43, January 2001.

[2] J. Heath, "Wires, switches, and wiring: A route toward a chemically assembled electronic nanocomputer," *Pure and Appl. Chem.*, vol. 72, no. 11, 2000.

[3] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K. Kim, and C. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, no. 5545, pp. 1313–1317, 2001.

[4] S. C. Goldstein and M. Budiu, "Nanofabrics: Spatial computing using molecular electronics," in *Proc. International Symposium on Computer Architecture (ISCA)*, July 2001, pp. 178–191.

[5] A. DeHon, "Array-based architecture for FET-based nanoscale electronics," *IEEE Trans. Nanotechnology*, vol. 2, no. 1, pp. 23–32, 2003.

[6] M. F. Jacome, C. He, G. de Veciana, and S. Bijansky, "Defect tolerant probabilistic design paradigm for nanotechnologies," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, 2004, pp. 596–601.

[7] A. DeHon, S. G. Goldstein, P. J. Kuekes, and P. Lincolnn, "Nonphotolithographic nanoscale memory density prospects," *IEEE Trans. Nanotechnology*, vol. 4, no. 2, pp. 215–228, 2005.

[8] G. Bourianoff, "The future of nanocomputing," *Computer Magazine*, pp. 44–49, Aug. 2003.

[9] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, pp. 1716–21, June 1998.

[10] M. Mishra and S. C. Goldstein, "Defect tolerance at the end of the roadmap," in *Proc. International Test Conference (ITC '03)*, 2003.

[11] "Sematech. International technology roadmap for semiconductors - 2004 update on emerging research devices." http://www.itrs.net/Common/2004Update/2004Update.htm.

[12] W. Maass, T. Natschlager, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[13] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks," *GMD-Report 148, German National Research Institute for Computer Science*, 2001.

[14] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with volterra series," *IEEE Trans. on Circuits and Systems*, vol. 32, pp. 1150–1161, 1985.

[15] H. Jaeger, "Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach," *GMD-Report 159, German National Research Institute for Computer Science*, 2002.

[16] W.Maass, T.Natschlager, and H.Markram, *Computational models for generic cortical microcircuits*. Chapman & Hall/CRC, Boca Raton, 2004, ch. 18, pp. 575–605.

[17] P. J. Courtois, *Decomposability: queueing and computer system applications*. Academic Press, 1977.

[18] T. Storm, "KiKS is a Khepera simulator," http://www.tstorm.se/projects/kiks/.

[19] W.Maass and H.Markram, "Theory of the computational function of microcircuit dynamics," in *The Interface between Neurons and Global Brain Function*, ser. Dahlem Workshop Report, S.Grillner and A. Graybiel, Eds. MIT Press, 2005, vol. 93.

[20] M. C. Ozturk, D. Xu, and J. C. Principe, "Analysis and design of echo state networks," *Neural Computation*, vol. 19, pp. 111–138, 2006.

[21] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[22] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication," *Science*, vol. 304, no. 2, pp. 78–80, April 2004.

[23] R. Lyon, "A computational model of filtering, detection and compression in the cochlea," in *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, May 1982.

[24] B. Bakker, F. Linaker, and J. Schmidhuber, "Reinforcement learning in partially observable mobile robot domains using unsupervised event extraction," 2002.

APPENDIX

Experimental methodology and benchmark tasks. The performance numbers reported in this paper for each benchmark task and machine configuration pair were derived by averaging the actual results obtained for a pool of 10 machines, each with randomly generated core(s) of the target size being considered, and using randomly selected training sets. The core sizes considered in the various experiments are explicitly indicated in the paper, except for those associated to the experiments reported in Figs. 3, 5, 6. In those, for each benchmark task we used the smallest core size(s) leading to performance saturation. For example, as can be seen in Fig. 8, for *Task 1* the 'saturation' core size is 200 nodes, since increases beyond that value lead to negligible performance improvements. Note further that the cores used in the experiments reported on the bottom of Fig. 6 have twice that of the baseline 'saturation' size. This is done to overcome the deleterious effects of structural noise on performance. Finally, all experiments (except, again, those reported on the bottom of Fig. 6) use cores with the same density of connectivity, namely, on average each core node has 2.7 input connections and 2.7 output connections. In turn, the experiments reported on the bottom of Fig. 6 use three times that density – once again, aiming at reducing the impact of structural noise on performance.

In what follows we provide a brief description of the benchmark tasks used in our experiments.

Domain 1: Wireless Communications. We considered two wireless channel models consisting of a sequence of linear filters of length 10, a time independent nonlinear transformation, and low amplitude additive white noise [21]. The task input is modeled as a sequence of equally likely random symbols from a possible set of 4, corrupted by the channel. The task attempts to recover the original sequence of symbols.

- **Task 1**: Channel equalization - using a basic channel model taken from the literature [22].
- **Task 2**: Channel equalization - using a modified (highly nonlinear) channel model. We substantially amplified the nonlinear distortion of the original model in [22], in order to increase the difficulty level of the task.

Domain 2: Voice. We selected three voice tasks, all of which are based on the TI46 corpus of inputs, consisting of 26 utterances of the digits 'zero' to 'nine', by 16 different speakers (8 men and 8 women), totaling more than 4000 inputs [16]. The inputs to the perturbation-based machine are first preprocessed using the Lyon Passive Ear model, which is a realistic model of the human inner ear[23]. Depending on the task, the output identifies the word or the gender of the speaker. We consider two different versions of those tasks, trained to recognize different sets of four words/digits, namely, 0-3 and 4-7, and the associated speaker's gender.

- **Task 3a**: Isolated word recognition: 0-3
- **Task 3b**: Isolated word recognition: 4-7
- **Task 4a**: Gender identification: 0-3
- **Task 4b**: Gender identification: 4-7

Domain 3: Robot Navigation/Control. Three benchmark navigation tasks were considered, all implemented on the Khepera robot [24]. This robot has two wheels with independently controlled speeds – each wheel's speed can be set to a discrete value between -10 to 10. When both wheels have the same speed, the robot moves along a straight line, when they have opposite speeds, the robot rotates in place, etc. Control decisions are made based on the readings of 16 noisy sensors placed on the periphery of the robot– 8 proximity infrared sensors and 8 directed ambient light sensors. The training set used for these experiments consists of a set of robot trajectories generated using a simple algorithmic script and a large number of random initial positions in the start region. The high accuracy MATLAB Khepera simulator KiKS v2.2.0 [18] was used to simulate the following three tasks:

- **Task 5**: Navigation on T-Maze - light at Position 1.The T shaped maze is shown in Fig. 10. The task is to navigate to end 1, if light is on, otherwise go to end 2. The task inputs and outputs are the 16 sensor readings and the speeds of the two wheels, respectively.
- **Task 6**: Navigation on T-Maze - light at Position 2. Same as Task 5, except for the position of the light (further away from the T junction)
- **Task 7**: Navigation on T-Maze - light at Position 3. Same as Task 6, except for the position of the light (even further from the T junction)

Domain 4: Vision/Motion Prediction. A single benchmark task is considered in this domain [16]. A moving object crosses an 8×8 field/array of sensors with at a random but constant speed and direction – with equal probability, the object is a ball or a bar [16].

- **Task 8**: Predict position of moving object. The task is to predict the readings of the inner 6×6 array of sensors, one step in advance. Task performance is measured using 2 metrics: errors associated with 'ON' sensor predictions and 'OFF' sensor predictions. The rationale for considering both types of predictions separately is that sensors are OFF most of the time, and thus an average across both errors would be too optimistic.

Domain 5: Parity Generation

- **Task 9**: Parity Generation 2W. The task is to generate a parity bit for a 2 bit sliding window on some input stream. The task input is the bit stream, and the output is the corresponding stream of generated parity values.
- **Task 10**: Parity Generation 3W. Same as Task 9, but considering now a 3 bit sliding window.