Using Randomly Assembled Networks for Computation*

Andrey Zykov 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 model as a promising choice for implementing next generation ubiquitous information applications on emerging nanotechnologies. Our argument centers on its suitability for technologies with low manufacturing precision, high defect densities and performance uncertainty. This paper discusses some of the possible advantages and pitfalls of this approach, and associated novel design principles.

Keywords: perturbation, computation model, embedded systems.

1 Introduction

Advances in the synthesis and self-assembly of nanoelectronic devices suggest that the ability to manufacture dense nanofabrics is on the near horizon, see e.g., [1,2]. Yet, effective ways of utilizing emerging nanoelectronic technologies still elude us. The tremendous increase in device density 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. 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. Thus, it is critical to consider and explore alternative computational models that can operate under such difficult conditions.

In this paper, we investigate a promising new class of computational model, called perturbation-based and show its potential to synergistically address the two sides of the complex system design equation: technology and applications. On one hand we argue it suitability in overcomming the reliability challenges associated with emerging nanoelectronics. On the other hand we focus on meeting the needs and leveraging the characteristics of emerging, but soon to be ubiquitous, soft real-time processing and control applications.

* This work is supported by the Gigascale Systems Research Center under the 'Alternative' Theme.

M. Cheng (Ed.): NanoNet 2008, LNICST 3, pp. 80–84, 2009.

2 Principles of Perturbation-Based Computing

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. Note this computational models was recently independently discovered by two research groups [3,4]. Yet, their work was driven by research pursuits and objectives quite different from those in this paper.

Perturbation-Based Machines. Fig. 1 symbolically depicts a perturbation-based machine M. As can be seen, it maps 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 [3]. A machine's D^M is thus a dynamical system realizing a very large pool of candidate operators, while the above mentioned 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. The output stage f^M maps the internal transient state to the desired output. Note that the precise internal dynamics of the core network need not have a specific form, as long as the core projects the input signals on a sufficiently rich set of basis functions.

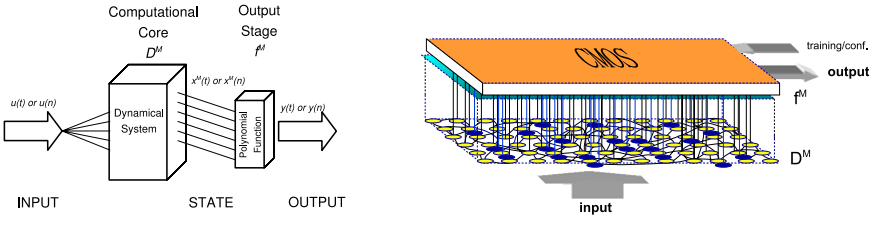

Fig. 1. On the left a Perturbation-based machine. On the right a proposed hybrid eNano-CMOS configurable platform for perturbation-based computing.

Universal Approximative Power and target applications. Based the work in [5], [3] 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. Still, although this 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 very high for certain tasks. The proposed computational model is inherently based on approximation. Therefore it is not expected to operate without errors, with perhaps the exception of approximation of very simple functions/operators. Although for some tasks this will be unacceptable, for others this presents an opportunity to tradeoff error rate against other costs, e.g., manufacturing cost, power consumption etc. An example of such tasks would be those involving real-time searches for opportunities, e.g., block matches across frames in video compression. If we miss an opportunity this will not cause algorithm failure, instead it results in a temporarily lower compression rate. Another class of applications involves systems with feedback, where such small/rare errors can be compensated subsequently through feedback overall having a negligible effect. More generally the aim is not to achieve high precision, but rather simplicity and universality. Blocks having moderate reliability can be bootstrapped to construct more complex and/or reliable operators, e.g., through averaging or other forms of aggregation [6] or specific mechanisms akin to the way complex logic functions are constructed from elementary logic gates (e.g. NAND, NOR, etc).

3 Design Principles for Perturbation Based Computing

We begin this section by first proposing a hybrid eNano-CMOS platform as a representative realization of perturbation-based machines. 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. 1 shows an abstract view of such a platform, with the key basis operators in the pool highlighted in bold. Clearly, this platform can directly leverage the formidable densities achieved by nanotechnologies to create computational cores of essentially arbitrary size. At the same time the more reliable CMOS layer allows to reliably configure (train) the output readouts to properly approximate the desired function. We discuss five underlying design principles next and refer the reader to [7] for experimental results supporting them:

1. Defect-Tolerance, Randomly Assembled Cores, and Training. First we need only ensure that the core and readout connectivity are sufficiently 'rich' to achieve the desired approximation after training. The designer needs only control the size and statistics of the core network without precisely specifying its topology. Manufacturing defects and heterogeneity in the network become part of its intrinsic randomness. This flexibility comes at the cost of performing a configuration/training step for each chip – which is indeed a costly requirement.

Yet it can be viewed as 'similar' to the overheads 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 re-synthesize the function to avoid defects. Defect mapping is typically done using test patterns that are either obtained/generated off chip or stored on chip. Re-synthesis involves reprogramming the function around the defects on the chip. In our case rather than defect mapping and re-synthesis 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 re-synthesis vs training is premature. Another potential disadvantage is possibility of excessive power consumption by random core in contrast to precisely controlled core. However reduced control requirements also open opportunities to make these random cores very cheap especially in terms of power consumption.

2. Fault-Tolerance Through Unstructured Redundancy. Second, fault tolerance can also be partially achieved by appropriately defining the statistics and size of the core. Intuitively, even if randomly assembled, a large dynamical network should incorporate sufficient redundancy to allow the readout layer to average out internal noise/soft errors. We refer to this as unstructured redundancy in the core, as it need not be explicitly designed, e.g., as would be the case with triple modular redundancy. Instead the designer need only decide on a sufficiently large core to address soft faults and/or internal performance variability, e.g., due to coupling etc. An obvious advantage of this approach is reduction in design cost in comparison to structured redundancy. A disadvantage this comes at a cost, bigger cores will consume more power.

3. Complete Core Sharing. Note that aside from general considerations on size and network statistics detailed core characteristics are task independent. Thus several different tasks that share the same input can in principle share its projection on the same core. For example word recognition and speaker identification tasks for the same speech input could share the same core. Such complete and parallel sharing of resources has the potential to substantially reduce overall system cost in terms of both area and power. Note however that the readout layer can not be easily shared across tasks, which may lead to a scalability problem if a core is to be shared among a large number of tasks.

4. Weakly Interconnected Networks and Spatial Decomposition. A potential problem with scaling to large cores is a scalability problem if random interconnections among nodes are necessary. We propose a fourth design principle towards overcoming this problem. The idea is that to introduce some hierarchy by only weakly interconnecting smaller cores. This allows one to control the interconnect costs as the size of the cores increase. Moreover this seems a natural way to randomly assemble cores, i.e., one where the primary form of connectivity is local. More generally one can imagine designs that leverage a large number of relatively small cores which serve as building blocks to create bigger cores as needed.

5. Nearly Decomposable Core Dynamics. The last design principle we propose relates to decomposition in terms of temporal dynamics. The idea is that

some applications are driven by (possibly coupled) dynamics at different time scales, which a designer might recognize and incorporate into his core design. For example a core design might include weakly interconnected cores operating at different speeds. One can imagine, creating cores with different response times to input signals, through some form of doping and/or processing. For applications exhibiting dependencies on multiple time scales such decompositions are very effective at reducing complexity. Furthermore purposefully combining fast and slow cores may present further advantages towards reducing power consumption. Note that the principle here is not perform careful core design, but simply define some large scale characteristics for connectivity and dynamics of its constituent subnetworks.

4 Conclusion

This paper explores an alternative computational model for "nanocomputing". We have pointed out a variety of interesting characteristics, such as the possibility of using randomly assembled nano network cores for computation. This approach is consistent with circumventing what appears to be an intrinsic difficulty and thus costly requirement of precisely controlling the manufacturing of nano systems. So the question is to what degree can one give up on precise control over manufacturing and still devise useful computation systems? In this paper we presented one possible approach but there may be others (perhaps more general) models.

References

1. Bourianoff, G.: The future of nanocomputing. Computer Magazine, 44–49 (August 2003)
2. Sematech. International technology roadmap for semiconductors - 2004 update on emerging research devices (2004), `http://www.itrs.net/Common/2004Update/2004Update.htm`
3. Maass, W., Natschlager, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002)
4. Jaeger, H.: The echo state approach to analysing and training recurrent neural networks. GMD-Report 148, German Nat. Res. Inst. Comp. Sci. (2001)
5. Boyd, S., Chua, L.: Fading memory and the problem of approximating nonlinear operators with volterra series. IEEE Trans. Circ. Sys. 32, 1150–1161 (1985)
6. Varatkar, G.V., Narayanan, S., Shanbhag, N., Jones, D.L.: Sensor network-on-chip. In: International Symposium on SOC (November 2007)
7. Zykov, A., Jacome, M., de Veciana, G.: Perturbation-based computing for next-generation embedded IT targeted at emerging nanoelectronics (submitted) (2008)