The Dissertation Committee for Andrey V. Zykov
certifies that this is the approved version of the following dissertation:

Exploring Scaling Limits and Computational Paradigms for Next Generation Embedded Systems

Committee:

Gustavo de Veciana, Supervisor

Nur Touba

Ross Baldick

Anirudh Devgan

Michael Orshansky

Exploring Scaling Limits and Computational Paradigms for Next Generation Embedded Systems

by

Andrey V. Zykov, B.S.; M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2009

To my family.

Acknowledgments

I wish to thank the multitudes of people who helped me. Time would fail me to tell of ...

Exploring Scaling Limits and Computational Paradigms for Next Generation Embedded Systems

Publication No. _____

Andrey V. Zykov, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Gustavo de Veciana

It is widely recognized that device and interconnect fabrics at the nanoscale will be characterized by a higher density of permanent defects and increased susceptibility to transient faults. This appears to be intrinsic to nanoscale regimes and fundamentally limits the eventual benefits of the increased device density, i.e., the overheads associated with achieving fault-tolerance may counter the benefits of increased device density – density-reliability tradeoff. At the same time, as devices scale down one can expect a higher proportion of area to be associated with interconnection, i.e., area is wire dominated. In this work we theoretically explore density-reliability tradeoffs in wire dominated integrated systems. We derive an area scaling model based on simple assumptions capturing the salient features of hierarchical design for high performance systems, along with first order assumptions on reliability, wire area, and wire length across hierarchical levels. We then evaluate overheads associated with using basic fault-tolerance techniques at different levels of the design hierarchy. This, albeit simplified model, allows us to tackle several interesting theoretical questions: (1) When does it make sense to use smaller less reliable devices? (2) At what scale of the design hierarchy should fault tolerance be applied in high performance integrated systems?

In the second part of this thesis we explore 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.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Trends in Development of Integrated Circuits

Over the last few decades the computational power of integrated circuits has been growing rapidly following to Moore's Law. This empirical law states that the number of transistors that can be placed inexpensively on an integrated circuit approximately doubles every two years [35]. Such growth has been achieved by miniaturization of devices, which not only enabled placement of more devices on the same chip area, but also made each device faster and less energy consuming. Indeed miniaturization has been an effective approach to increase computational power. Unfortunately this approach has fundamental limits and these limits are at this point not far away. A device in the traditional sense cannot be smaller than one atom. In fact it is problematic to approach this limit. Note that modern lithography is already at the nano scale with feature sizes 35nm-45nm, while the silicon crystal lattice spacing is 0.54nm. On the one hand, manufacturing becomes increasingly expensive making such levels of miniaturization impractical. On the another hand, smaller devices become more prone to defects and soft errors, which must be overcome through testing/defect mapping and soft faults tolerance redundancy. The overheads associated with testing and redundancy may very quickly exceed the benefits associated with reduced device's size.

Another problem is power consumption. Although smaller devices tend to be more energy efficient, an increased number of devices per chip still requires more power and results in a higher power density. Also at current levels of miniaturization the energy benefits of smaller devices are partially compensated by the increased role of wires as the load for a driving device. For this reason the operational speed of

devices does not increase as quickly as it could given the intrinsic speedup of devices with reduced size. In any case it appears difficult to substantially reduce the device size beyond the current level. One possible way to achieve higher computational power is to use even more devices (on several chips or on a chip with larger area). Such systems would consume proportionally more power. The power budget is however limited by heat conductance capability and the maximum temperature at which the systems can operate. Although it is possible to enhance heat conductance to some extent, it cannot be made arbitrarily high, so power budget cannot be substantially increased.

Fortunately it is possible to do more computation using the same energy. This becomes clear when we consider how power gets dissipated during switching (i.e., dynamic power dissipation). When, for example, a line at a low level has to be switched to a high level, the pull up transistor starts conducting current from the power line to the signal line and the voltage across the transistor channel roughly equals that of the power supply. As we know power dissipation inside the channel is $I * V$ - product of the current flowing through the channel and the voltage across the channel. The total energy dissipated in the channel is the integral of the power which equals $Q * V/2$ (assuming a constant capacitance) - i.e., the product of the total charge passed through the channel by half of voltage swing. Noting that $Q = C * V$, where C is the capacitance of the line, the total energy dissipated in the channel is $C * V^2/2$, which gives the well known formula for dynamic power dissipation $f * C * V^2/2$, where f is switching frequency of the line.

The switching here happens in a very aggressive manner where voltage across the conducting channel is big. This helps to switch in the fastest possible way, but it also costs a lot of power. Note that power depends quadratically on the voltage. So, for example, if we reduce the voltage, switching will happen more slowly (roughly linear with voltage), but power will be reduced quadratically. So this is how one can achieve significantly higher computational capability under a power constraint. One has to use a larger number of devices operating in a slow low power mode.

The number of devices grows quadratically with desired increase in computational capacity, while the speed of each device drops linearly and power consumption per device drops quadratically.

Of course the physics of silicon transistors will not scale in this idealized manner all the way down to zero voltage. A potential solution is to use alternative materials that will allow operation at extremely low voltages. Another potential solution is called adiabatic switching [13, 20] which allows the use silicon transistors operating at a high supply voltage. However switching happens in a not so aggressive way as it happens in the mode described above. In this case the voltage of the power line is not constant, but it increases gradually when charging the signal line. In this way the voltage across the conducting transistor channel is small all the time, but the transferred charge is the same, so the energy per switch becomes small. Of course the current through the channel is small too, so switching takes longer.

The bottom line is that in order to achieve significantly higher computational capability under power constraints, one needs to increase quadratically the number of devices. To make this practical, the system size and cost have to be reasonable, unless we try to build a supercomputer where size and cost may be a lesser factor. However even for a supercomputer power consumption must be within reasonable limits. An example of a such supercomputer is the 'Green Flash' project [43, 44] that basically adopts the above mentioned principle of using an increased number of low-power mobile chips instead of general-purpose processors to achieve significant (about 60 times) power reduction when huge computational power is needed for climate modeling. In this case the system size is not as important as in the case of an embedded system, so regular low power chips work sufficiently well.

A straight forward approach could be to use many chips, but this would require significant system size, what is not acceptable for embedded systems, and the cost would grow proportionally to the number of chips. An additional problem with this approach is the large number of macro connections among these chips. So obviously a new technology is required that could provide a much larger number

of devices than current technology, though not necessary as small devices as in the current technology. However the cost of such systems should not be very high and the size should be reasonable. Apparently in order to achieve reasonable system size, one will have to consider packing devices tightly in a 3D volume, in contrast to current technology where devices are packed inside a thin layer on fairly small chip areas.

True 3D integration is still not a realistic technology in the near future. However it is not required to achieve partial success. For example, one can use traditional photolithography to imprint circuits on a large area of a flexible thin film. This thin film can then be compacted within a 3D volume. This way a tight 3D packing can be achieved using essentially a 2D technology. Such a technology would allow us to extend the life of Moore's law beyond the limit of atomic size for planar systems.

Systems built according to above mentioned principals would operate quite differently from current systems. The system size would be much larger and at the same time each element of the system would operate at a low speed. Naturally two questions arise here. One question is what implications these properties have on current design approaches if they were applied to prospective systems. A second question is what alternative computation models would be well suited to these prospective technologies.

It is interesting and inspiring to compare engineered computational systems with the human brain. The latter is a computational system which is particularly well suited for signal processing and control tasks. The membrane potential in neurons is a few dozens of millivolts, while the voltage swing in microchips is a few hundreds of millivolts (used to be thousands). Switching frequency in neurons is roughly 100Hz, while in current microchips it is a few GHz. The number of synapses in the human brain is 10^{14}, while the number of transistors on a chip is 10^9 (note here that if computation capacity is affected by frequency, memory capacity depends only on device count). Neurons are not as small as today transistors, but very large

amounts of them are packed in 3D volumes, while transistors occupy thin layers of planar chips.

It is quite possible that new computational models suitable for new prospective technologies will not be as efficient for all computational tasks for which old technologies were efficient. However one can hypothesize that they can be efficient for at least the tasks where the human brain is efficient - cognitive real time signal processing and intelligent control.

This thesis addresses two aspects associated with above mentioned challenges. On one hand, how the scaling of technology affects traditional design approaches. On the other hand, we argue that it may be beneficial to use a new computational paradigms and associated design principles under new scaled technology that might be suitable for future embedded systems. I discuss these two themes in more detail next.

1.2 Scaling Limits under Traditional Design Approach

It is widely recognized that device and interconnect fabrics at the nanoscale will be characterized by a higher density of permanent defects and increased susceptibility to transient faults. This appears to be intrinsic to nanoscale regimes and fundamentally limits the eventual benefits of the increased device density, i.e., the overheads associated with achieving fault-tolerance may counter the benefits of increased device density – density-reliability tradeoff. At the same time, as devices scale down one can expect a higher proportion of area to be associated with interconnection, i.e., area is wire dominated. In our work we theoretically explore density-reliability tradeoffs in wire dominated integrated systems. I derive an area scaling model based on simple assumptions capturing the salient features of hierarchical design for high performance systems, along with first order assumptions on reliability, wire area, and wire length across hierarchical levels. I then evaluate overheads associated with using basic fault-tolerance techniques at different levels

of the design hierarchy. This, albeit simplified model, allows me to tackle several interesting theoretical questions:

- When does it make sense to use smaller less reliable devices?
- At what scale of the design hierarchy should fault tolerance be applied in high performance integrated systems?

I then introduce a more general reliability model including devices and wire failures, and briefly discuss regimes where it might be beneficial to jointly address defect and fault-tolerance. My analysis reveals two critical parameters – technology and design scaling factors – which are key to predicting the reliability requirements for emerging technologies if traditional hierarchical design continues to be used.

1.3 Alternative Computation Model

In the second part of this thesis I explore perturbation-based computational models as a promising choice for implementing next generation ubiquitous information technology on unreliable nanotechnologies. I 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. I 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. I 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. My results provide strong evidence for the relevance and potential of this class of computational models when targeted at emerging unreliable nanoelectronics.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 introduces an area scaling model in the wire dominated case relevant for large dense systems. It also presents an analysis of the impact of this scaling model on redundancy that should be used to achieve reliability in such systems. Chapter 3 introduces an alternative computational model called Perturbation Based Computing and presents design principles for this model. Finally, Chapter 4 provides conclusions, a summary of the dissertation contributions, and it makes suggestions for future research.

Chapter 2

Exploring Traditional Design Approach in the Wire Dominated Regime

2.1 Introduction

Future integrated systems will be implemented on increasingly dense substrates. These in turn are expected to be characterized by high densities of manufacturing defects and high rates for transient faults [1, 5, 23, 34]. In order to achieve a desired manufacturing yield and system reliability, engineers apply defect- and fault-tolerance techniques. These techniques in turn incur overheads associated with additional circuitry and redundancy [39]. Such overheads take up area on the chip and thus increase power consumption. Thus, it is possible that the defect- and fault-tolerance overheads may grow faster than the extra area afforded by the increased density of devices and wires the new technologies provide. This can happen if the reliability of the substrate drops quickly with increasing density. As part of this thesis I develop a theoretical model to study when this is indeed the case. As mentioned earlier in addition to these reliability challenges, limits on the acceptable density of power dissipation may also constraint what is achievable with these new technologies.

Another key observation is that as technology scales down, the impact on area and power consumption of wires grows faster than that of devices (at least for high performance systems). This is supported empirically by Rent's Rule [7, 8, 14, 31]. It states that the number of external wires for a circuit sub-block is proportional to the size of this sub-block (e.g., in gates) to the power r, where r is referred to as Rent's exponent and is typically about 0.6-0.7 for high performance systems.

Rent's Rule can be viewed as arising from consistency requirements across levels of a design hierarchy – i.e., reflects a design style based on interconnection of increasingly complex sub-blocks. The exponent may vary from system to system (or sometimes even within a given system at different levels of the design hierarchy) reflecting not only design style, but also, the type of functionality being implemented. When Rent's exponent is greater than 0.5 for a 2D circuit, then the density of wires increases with system size provided that gates are tightly packed. In practice, this is reflected by an increase in the number of metallization layers used for chips over the last decades – from 1 to 10 or more layers in today modern chips. However it is technologically problematic to continue increasing the number of layers, so at some point it will become impossible to pack devices tightly. At that point the chip's area will be wire dominated.

In such a regime (dynamic) power consumption may also be wire dominated. Indeed, static power consumption depends mostly on device physics, while dynamic power consumption is roughly proportional to load capacitance which is already dominated by wires in modern technologies. For example [47] considers a 3D design where devices were vertically stacked in 4 planes and show that the major effect is a reduction in wire length accompanied by a dramatic reduction in power consumption. To understand the fundamental characteristics of future technologies, one must properly reflect wires' increasingly dominant impact on area, power and even performance.

Related work. In this thesis I combine a novel scaling model (capturing the wire dominated regimes of interest) with traditional reliability analysis. This might be viewed in contrast to research initiated by [42] tackling computability with unreliable devices, but ignoring device and wiring overheads, see e.g., [4, 15, 18, 38]. By considering wire dominated regimes my work also differs from previous work considering reliability and overhead models based solely on gate count see e.g., [19, 30, 36]. In contrast to previous work [7, 24], the scaling model itself focuses on the wire dominated regime.

Contribution. In this chapter I consider substrate technologies that are capable of delivering a high density of devices and wires but have higher defect densities and transient fault rates. I consider the case where such a substrate is used to implement a 'monolithic' high performance system. I start by proposing and analyzing an area scaling model for such technologies based on several natural assumptions. The model exhibits how area grows with the complexity of the system (number of gates) in a wire dominated regime. This is my first contribution.

Using this scaling model I consider the overheads associated with achieving fault-tolerance by applying spatial redundancy at different levels of the system hierarchy. This simple model enables me to address two questions, which are the second key contribution of this chapter:

1. when do smaller, but less reliable devices make sense,

2. and at what level of the design hierarchy should fault tolerance be applied?

By combining a novel scaling model (capturing the wire dominated regime of interest) with traditional reliability analysis to tackle these questions.

The final contribution includes a brief discussion of my wire scaling result in the context of defect-tolerance and whether joint consideration of defect- and fault-tolerance problems might be beneficial. Indeed traditionally these have been considered separately. Defect tolerance is characterized by yield, and testing procedures are very important to guarantee a very low level of defective systems is shipped to customers. Soft fault tolerance is characterized by system reliability and intended to guarantee a very low rate of system failure. If these are addressed simultaneously, then soft tolerance techniques could in principle be used to design systems which tolerate some density of defects making them undetectable during testing. However, such defects would compromise the system's subsequent tolerance to soft-errors, i.e., system failures due to soft-errors may occur more frequently than was intended by

design. Therefore simultaneous use of defect and fault tolerance will create a complex system reliability distribution instead of simple yield function, where systems are either good or faulty. This discussion is supplemented by a consideration of the implications of my work with respect to power-reliability tradeoffs and implications for the development of future computation technologies based on the scaling results derived in this thesis.

Organization. In Section 2.2 I propose and analyze an area scaling model for considered technologies based on several natural assumptions. The model exhibits how area grows with the complexity of the system (number of gates) in a wire dominated regime. Using this scaling model I consider the overheads associated with achieving fault-tolerance by applying spatial redundancy at different levels of system hierarchy. Section 2.2 of this chapter focuses only on gate reliability, while Section 2.3 motivates a general model where both gates and wires may fail. This permits me to consider the manner in which device vs wire reliability impact the usefulness of a given technology. In Section 2.4 I briefly discuss my wire scaling result in the context of defect-tolerance and whether joint consideration of defect- and fault-tolerance problems might be beneficial. Section 2.5 includes a discussion of the implications of my work with respect to considering power-reliability tradeoffs. Section 2.6 highlights some possibilities for development of future computation technologies based on the scaling results derived in this chapter. Finally Section 2.7 offers some closing comments and perspective for this work.

2.2 Scaling Model and Basic Reliability Analysis

This section presents a novel area scaling model, capturing the wire dominated regime, which is then used to evaluate the density-reliability tradeoffs. I begin by carefully introducing several natural assumptions for the underlying model.

2.2.1 Wire Dominated Area Scaling

The first assumption concerns interconnection across hierarchical levels. Traditional hierarchical design approaches to building increasingly complex systems are based on interconnecting sub-blocks. For example a pipelined CPU is realized based on blocks such as a fetch instruction stage, decode instruction stage, execute stage, registers file, external memory block, etc. The execute stage is itself built using different functional blocks (e.g., full word adders, multipliers, etc), where each block is built out of smaller blocks (e.g., one bit adders, etc). As a result when such systems are implemented on a substrate they lack structural regularity across hierarchical levels. By contrast, for intrinsically regular functions (e.g., memory arrays, FPGAs) one can adopt a more flat design style where the system is comprised of a large number of simple blocks. The implementation of such systems might eventually reflect regularity in placement and routing. In this chapter I focus on hierarchical designs whose eventual implementations on a substrate would exhibit 'irregular' routing and placement across levels of the hierarchy.

Assumption 2.2.1. ***(Hierarchical consistency)*** *Consider systems designed in a hierarchical manner across multiple levels. Hierarchical consistency in interconnecting sub-blocks at different levels means that Rent's Rule should apply. Specifically,*

$$N_{ext}(M) = k_w M^r,$$

where $N_{ext}(M)$ is number of external wires for a block with M gates (or sub-blocks) and r is Rent's exponent (typically 0.6-0.7), k_w is a proportionality constant relating external wires to number of gates to the r^{th} power.

Following [16] I refer to Rent's Rule as satisfying *hierarchical consistency*. Indeed, consider creating a block by composing P sub-blocks each comprised of M gates. By Rent's rule each sub-block has $N_{ext}(M)$ external wires and the number of external wires for the larger block should be $N_{ext}(M)P^r$. Yet the larger block has a

total of MP gates, hence the total number of external wires should also be given by

$$N_{ext}(MP) = k_w(MP)^r = (k_w M^r)P^r = N_{ext}(M)P^r,$$

which exhibits the above mentioned hierarchical consistency. Note that Rent's Rule deals with logical wires, i.e., abstract connections among blocks [8, 31]. These logical wires may be implemented using one or more physical wires. So, for example, repeaters may be inserted in implementing a logical wire subdividing it into several physical wires. The area cost of such a logical wire will be defined as its constituent physical wires and devices used to realize it. This leads me to a second key assumption.

Assumption 2.2.2. ***(Wire area)*** *Assume a block's area is the sum of its constituent gates and wires. The area of a wire is assumed to be proportional to its length, i.e.,*

$$A_w(l) = k_l l,$$

where $A_w(l)$ denotes the area of a wire of length l and k_l is a proportionality constant.

Length will be measured in linear minimal gate sizes, i.e., the linear minimal gate size l_g is 1. Similarly area is measured in minimal gate areas, so that minimal area of a gate is $a_g = l_g^2 = 1$. In these units the k_l reflects average area per unit length wire in units of minimal gate area. Note however that a chip may have several metal layers that would result in a smaller coefficient k_l, e.g., 10 metal layers at best gives 10 times the area to route wires, reducing the coefficient by a factor of 10.

In general Assumption 2.2.2 is expected to be reasonable. A wire's area is unlikely to grow sub-linearly in its length. In some cases it may grow super-linearly, e.g., if high performance is required, extra wide wires may be used to reduce resistance or extra repeaters to reduce latency. One can expect such wires to be only a small fraction which are on critical paths, and thus they would not significantly impact the overall scaling of area.

The third assumption reflects my focus on hierarchically designed systems, which when mapped onto substrates exhibit irregular routing and placement.

Assumption 2.2.3. ***(Irregular routing)*** *The average length of wires used to interconnect sub-blocks having area A is proportional to their linear size, i.e.,*

$$L_w(A) = k_r\sqrt{A},$$

where $L_w(A)$ is the average length of wires interconnecting blocks having area A, and k_r is a proportionality constant reflecting the design's characteristics.

Note that interconnecting wires at a given scale, i.e., interconnecting blocks of a given size A, may have varying length, i.e., some may be short. Through Assumption 2.2.3 I posit that for systems which are hierarchically designed, resulting in irregular routing and placement, one should still expect the *average* length of such interconnections to be on the order of the linear size of the blocks they interconnect.

With these three assumptions in place one can show an area scaling law in system complexity (number of gates) capturing dominant role of wires on the area.

Theorem 2.2.4. ***(Wire dominated area scaling)*** *Under Assumptions 2.1-2.3 the growth in area A with system complexity M (in gates) satisfies the following differential equation:*

$$dA = \frac{A}{M}dM \;+\; k_l(k_r\sqrt{A})(k_w(1-r)M^{r-1}dM). \tag{2.1}$$

The solution to this equation for $r \neq 0.5$ is given by

$$\begin{aligned} A(M) &= a_g(\sqrt{M} + tdM^r)^2 \\ &= a_g(M + 2tdM^{r+0.5} + (td)^2M^{2r}), \end{aligned} \tag{2.2}$$

where $t = \frac{k_l k_w}{\sqrt{a_g}}$ is referred to as the technology scaling factor while $d = k_r\frac{1-r}{2r-1}$ is a design scaling factor.

I sketch the proof for Theorem 2.2.4 as follows. The differential growth in area represented by Eq. 1 has two terms on the right hand side:

$$dA = \underbrace{\frac{A}{M}dM}_{\text{Term 1}} \quad + \quad \underbrace{k_l(k_r\sqrt{A})(k_w(1-r)M^{r-1}dM)}_{\text{Term 2}}.$$

The first term can be interpreted as follows. Consider a block of area A with M gates, then the area per gate and internal wires for such a block is A/M thus if additional dM gates are added to create larger blocks, area should grow proportionally to A/M. The second term represents additional area associated with wires interconnecting blocks of size M. Consider a block of size M_2 consisting of M_2/M_1 sub-blocks of size M_1. By Rent's Rule the total number of external wires for all blocks of size M_1 is $\frac{M_2}{M_1}N_{ext}(M_1)$. This includes *some* of the internal and *all* of the external wires for the block of size M_2. However by Rent's Rule the number of external wires of the larger block is $N_{ext}(M_2)$, so the number of wires used to interconnect blocks of size M_1 within M_2 is $\frac{M_2}{M_1}N_{ext}(M_1) - N_{ext}(M_2)$. By hierarchical consistency and letting $M_2 = M + dM$ and $M_1 = M$ I obtain a differential number of interconnecting wires for blocks of size M in the form

$$\frac{M+dM}{M}N_{ext}(M) - N_{ext}(M+dM).$$

This can be evaluated using Rent's formula. The second term also reflects my assumptions on the length and area of such wires, i.e., Assumptions 2.2 and 2.3. The solution Eq. 2 can be easily checked by substitution. Note that as expected the growth in area $A(M)$ includes a linear term in the number of gates, as well as other terms which reflect the dominant role of wires and whose growth is faster than linear.

Two key scaling parameters emerge. The first, called the *technology scaling factor*, depends on the average number of wires per gate k_w and wire length per linear gate length $k_l/\sqrt{a_g}$. The second, referred to as the *design scaling factor*, depends solely on characteristics of the design, i.e., on Rent's exponent r and k_r the

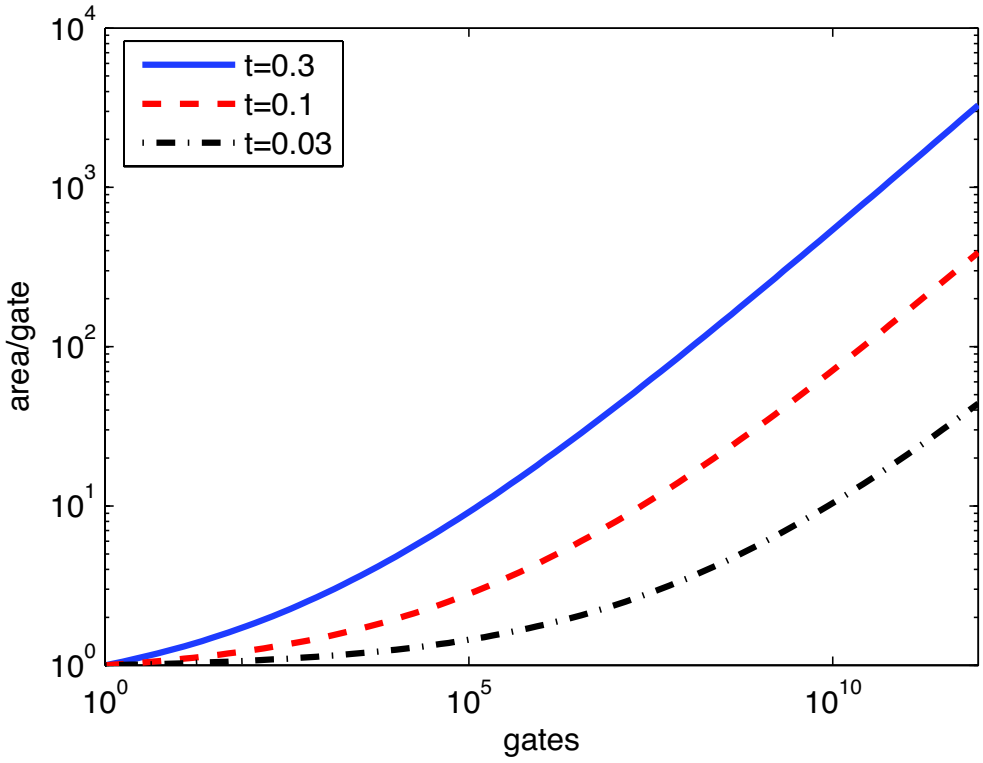

Figure 2.1: The area $A(M)/M$ as a function of M for various technology factors and $d = 1$.

parameter capturing the length of wires interconnecting blocks of similar size. The graph in Fig. 2.1 exhibits the growth in area per gate, i.e., $A(M)/M$ for $d = 1$ for different technology scaling parameters; $t = 0.1$, might be viewed as a baseline where $k_w = \sqrt{a_g}$, i.e., wire width is the same as minimal linear gate size and $k_l = 0.1$, e.g., 10 or so packed metallic layers for wiring.

It is interesting to compare wire scaling model (Eq. 2.2) with real technology trends as feature size decreases. In modern systems increased connectivity is achieved by adding extra metal levels. Figure 2.2 depicts the number of metal levels vs. metal 1 pitch as this pitch decreases over years. The solid line corresponds to actual data obtained from a combination of ITRS reports for years 2000–2008. Before 2009 it is historical data, after year 2009 the data corresponds to ITRS predictions. The dashed line corresponds to my wire scaling model (Eq. 2.2) assuming that de-

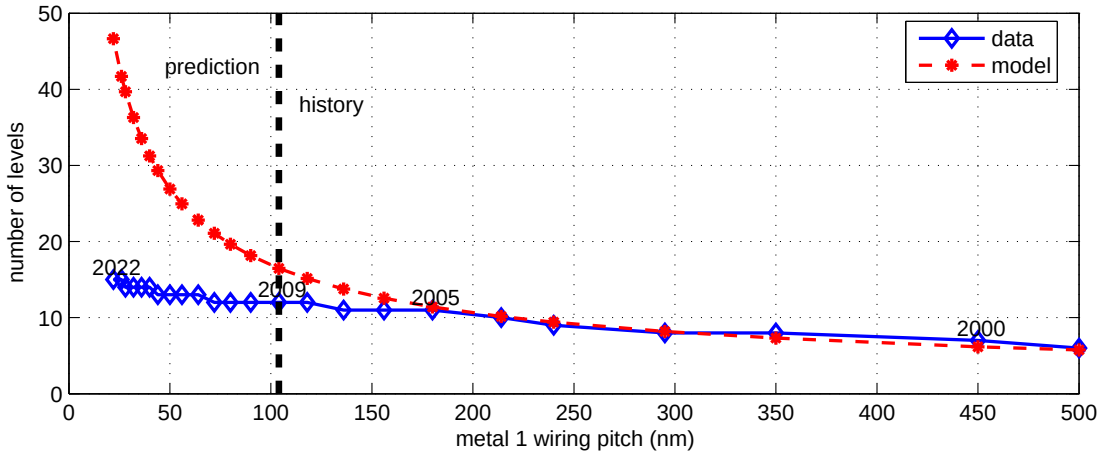

Figure 2.2: Number of metal levels vs. metal 1 wiring pitch according to ITRS historical and prediction data compared to wire scaling model (Eq. 2.2).

vices are tightly packed with some fixed packing density. On this graph we can see a pretty good match between actual data and the model until year 2005 reflecting the fact that extra metal levels were capable to provide the required connectivity. But starting year 2005 the number of metal levels begin to lag behind the number suggested by my model and this deviation grows dramatically when compared to the ITRS predictions. I conjecture that the deviation between my model and the true data between 2005-2009, caused by an unwillingness to increase the number of (costly) metal layers, resulting in a decrease in the device packing density. Recall that my wire scaling model plotted here assumes a fixed packing density.

2.2.2 System Reliability and Fault-tolerance Overheads

To evaluate density-reliability tradeoffs, we need to characterize fault-tolerance overheads. The simplest way to achieve this via n-way spatial redundancy, i.e., replicate an unreliable sub-block n times and introduce bitwise majority voting to obtain reliable outputs. Assume voters are reliable and have negligible area requirements this approach achieves an exponential (in n) improvement in reliability with a linear (in n) area overhead – I shall reconsider this in the sequel. Clearly there are many alternatives to achieve fault-tolerance. For example, temporal redundancy

requires much less overhead, but can only be applied at a sufficiently high architectural level (i.e., allowing "rollback") with blocks having sufficiently high reliability. My motivation here is to consider high-performance, computation and/or control functions (e.g., those required to implement temporal redundancy) where spatial redundancy is a reasonable approach to achieve a significant boost reliability. I make the following assumption.

Assumption 2.2.5. ***(Reliability and hierarchy)*** *The following assumptions concerning the reliability of components and application of n-way redundancy on a continuum of levels of the design hierarchy are made:*

a. *The probability of failure of a system is the sum of the failure probability of its constituent blocks. Thus a block of size M_0 gates has a probability of failure $p(M_0) = M_0 p_g$ where p_g is the probability of failure of a gate. Wires and voters are assumed to be reliable for now.*

b. *Spatial n-way redundancy is used to enhance a system's reliability. For a system of total size M_S gates, I assume redundancy can be applied at any of a continuum of hierarchical levels, indexed by the size of the blocks M_0, where M_0 can range from 1 to M_S gates.*

Assumption 2.2.5.a can be viewed as *consistency* assumption where failure probabilities are additive across constituent sub-blocks and scales. This corresponds to focusing on a regime where the failure probabilities are fairly low, and the probability of failure of a block of size M_0 gates is linear

$$p(M_0) = 1 - (1 - p_g)^{M_0} \approx M_0 p_g$$

if higher order terms can be ignored.

Assumption 2.2.5.b means that one may apply spatial redundancy to blocks of any size. In practice this would not be possible, but this idealization allows one to

roughly investigate the granularity at which spatial redundancy should be applied. Specifically, if n-way redundancy is applied across blocks of size M_0 gates the system would have M_S/M_0 sets of redundant blocks. Then for $n = 3, 5, 7 \ldots$ the probability of failure of an n-way redundant block of size M_0 is given by

$$\sum_{i=\frac{n+1}{2}}^{n} \binom{n}{i} (1 - M_0 p_g)^{n-i} (M_0 p_g)^i \approx \binom{n}{\frac{n+1}{2}} (M_0 p_g)^{\frac{n+1}{2}}.$$

Finally using Assumption 2.2.5.a the probability of failure for the overall system P_S composed of $\frac{M_S}{M_0}$ such blocks is $P_S = \frac{M_S}{M_0} \times \binom{n}{\frac{n+1}{2}} (M_0 p_g)^{\frac{n+1}{2}}$.

Ignoring voters and associated circuitry, and irrespective of the block granularity M_0 at which n-way spatial redundancy is applied the overall number of gates in the system increases by a factor of n. However if replication occurs at lower levels of the hierarchy, longer wires will be required at higher levels of the design hierarchy. Indeed these wires not only get replicated n times, but also become longer taking even more area. So the total area overhead of realizing n-way spatial redundancy will be higher if it is realized at a lower level of the design hierarchy.

To properly capture the overheads when n-way spatial redundancy is applied starting at a hierarchical level M_0 I modify Eq. 1 to reflect the redundancy overheads. For $M \leq M_0$ it remains the same which by Eq. 2 gives an area $A(M_0)$ for a block of size M_0. For $M > M_0$ this is modified as follows. The initial condition becomes $M = M_0$. The initial area with n-way redundancy at scale M_0 is $nA(M_0)$. The differential growth in area for the system with n-way redundancy and $M > M_0$ is now given by

$$dA = \frac{A}{M} dM \; + \; n k_l (k_r \sqrt{A}) (k_w (1 - r) M^{r-1} dM. \tag{2.3}$$

This can be viewed as multiplying the design scaling factor by n to capture the additional overhead associated with redundant wires. One can thus solve for the area and reliability of the system when n-way redundancy is applied at scale M_0 of the design hierarchy, giving the following result.

Proposition 2.2.6. ***(Area-reliability tradeoff)*** *Under Assumptions 2.1-2.3 and 2.5 the total system area A_S and system reliability P_S for a system with complexity M_S when n-way redundancy is applied at scale M_0 of the design hierarchy are given by:*

$$\begin{aligned} A_S(M_0) &= n a_g \frac{(\sqrt{M_0} + tdM_0^r)^2}{(\sqrt{M_0} + ntdM_0^r)^2} (\sqrt{M_S} + ntdM_S^r)^2 \\ P_S(M_0) &= \frac{M_S}{M_0} \times \binom{n}{\frac{n+1}{2}} (M_0 p_g)^{\frac{n+1}{2}}. \end{aligned}$$

These relationships allow one study the tradeoff between area A_S and reliability P_S as one varies M_0. Note however that M_0 cannot take arbitrary values – it should lie in the range from 1 to M_S and further for the region near 1 should not be considered as it corresponds to applying redundancy at the level of a single gate.

The graph in Fig. 2.3 shows both the area per gate and the overall system reliability when 3-way redundancy is applied to blocks M_0 ranging from a single gate to the overall system size $M_S = 10^{12}$ for a fixed probability of gate failure $p_g = 10^{-14}$. As can be seen, if redundancy is applied at a higher level M_0 one sees a lower area overhead but also a lower reliability. Thus there is a highest scale M_0 at which one can apply redundancy to achieve a given overall system probability of failure P_S. This is exhibited graphically on the plot.

Using this model I can consider if it is worth moving to smaller less reliable gates. Consider a system of fixed complexity (number of gates without redundancy) $M_S = 10^{12}$ to be implemented on a fixed absolute area, with the fixed acceptable overall probability of failure $P_S = 10^{-16}$. Given we are using n-way redundancy, we can ask what is the maximum acceptable probability of gate failure p_g such that the overheads associated with reaching the desired P_S fit in the absolute area of interest. As we reduce gate size (area) a_g, i.e., increase the density of a technology, we expect to be able to afford higher overheads for fault-tolerance, allowing higher probabilities of gate failure. The plot in the Fig. 2.4 exhibits curves for the maximum tolerable probability of failure for different gate sizes and different degrees

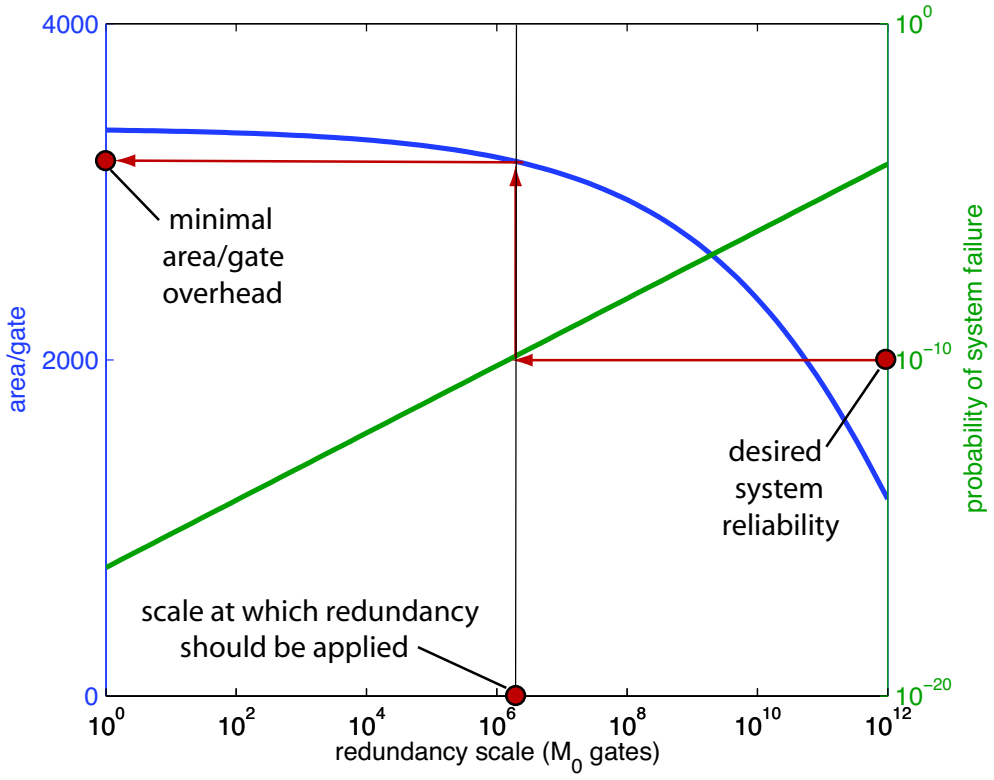

Figure 2.3: System overhead in area/gate and reliability as the hierarchical level at which redundancy is applied varies.

of n-way redundancy. The horizontal axis on this graph corresponds to the size of gates measured with respect to size of bigger gates of area a_{g0} that would give the same total system area if no redundancy were applied. Such big gates should have probability of failure p_g at most $P_S/M_S = 10^{-28}$ to meet a target system reliability P_S. Note that the exhibited curves have a finite domain representing what is possible when M_0 ranges from 1 to M_S.

These curves reflect limits on the reliability of gates, i.e., where the redundancy overheads to achieve the overall system reliability consumes all extra area afforded by reduced gate size. All points below (better system reliability) and left (less area) from any point of these curves are acceptable. Points which are to the right or above all these curves are unacceptable, because a system built using

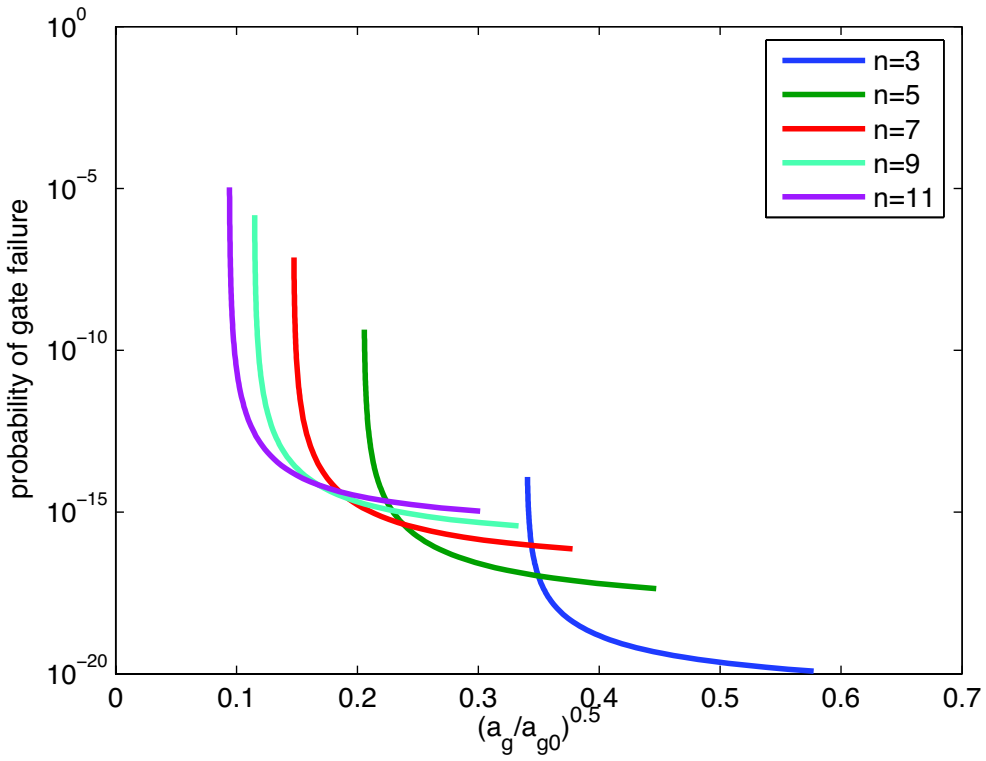

Figure 2.4: Minimum affordable gate reliability for reduced device size.

smaller less reliable gates will occupy more area than non-redundant system built using larger more reliable gates. The plot in Fig. 2.5 shows the acceptable region across various levels of spatial redundancy.

2.3 Generalized Scaling Model

The results obtained in the previous section were predicated on both voters and wires being perfectly reliable. Let me reconsider these in turn. First I assumed voters can be considered to be perfectly reliable at no cost. This is reasonable if redundancy is not applied at a very low level in the hierarchy. In this case each block's cost will be orders of magnitude higher than the cost of voters. Indeed, the complexity of single bit voter is very small. Also the number of voters is proportional to the number of external wires which by Rent's Rule scales as M^r ($r \approx 0.6 - 0.7$)

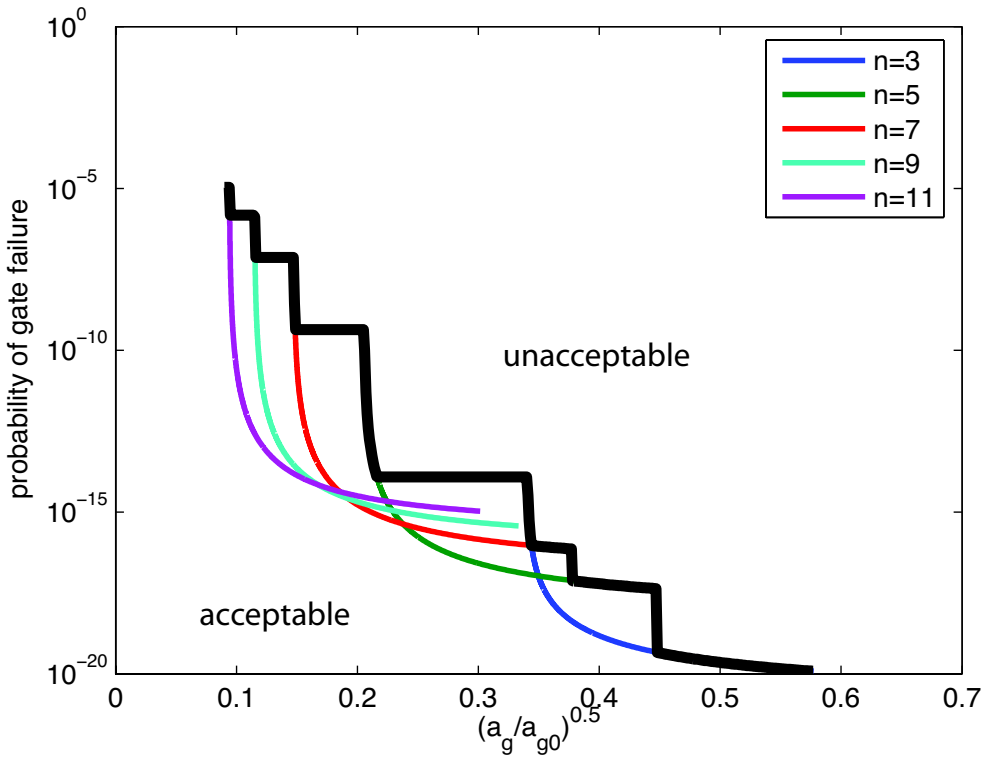

Figure 2.5: Minimum affordable gate reliability for reduced device size (outline).

which is small relative to M. Thus one could in principle use 'big' or more reliable devices to implement voters at a negligible area cost. The second assumption that wires are reliable is harder to justify. On one hand it is likely that ionizing particles, as a source of soft faults, are more likely to impact active device areas than signals across wires. However it is not clear whether wires in emerging technologies might not also be vulnerable to ionizing particles. If this were the case, then a reasonable model would be a probability of wire failure which is proportional to its length (or equivalently its area). On the other hand it is widely recognized that other internal sources of transient errors are of critical concern. For example coupling among wires is data dependent and might be modeled as a probability of failure which is proportional to wire length. Also, delay variability, in some cases may be

data dependent [1] and might again be modeled as a random event. Though in this case, it is not clear that the probability of failure is proportional to wire length, it is true that a longer wire would have a higher probability of failure. To better capture concerns with the reliability of wires and investigate their importance relative to gate reliability I revise Assumption 2.2.5 as follows.

Assumption 2.3.1. ***(Generalized reliability and hierarchy)*** *The following assumptions concerning the reliability of wires and components and application of n-way redundancy on a continuum of levels of the design hierarchy are in effect:*

a. *The probability of failure of a system is the sum of the failure probability of its constituent blocks. A block of size M_0 gates and total wire length L_0 has a probability of failure $p(M_0) = M_0 p_g + L_0 p_w$ where p_g is the probability of failure of a gate and p_w is the probability of failure of a wire per unit length. Voters are assumed to be reliable.*

b. *Spatial n-way redundancy is used to enhance a system's reliability. For a system of total size M_S gates, I assume redundancy can be applied at any of a continuum of hierarchical levels, indexed by the size of the blocks M_0 where M_0 can range from 1 to M_S gates.*

c. *When redundancy is applied at level M_0 I assume that 'long' wires, i.e., interconnecting blocks of size M_0 or above are made reliable but have greater area per unit length by factor k_o.*

The key idea underlying this assumption is as follows. When spatial redundancy is applied at a certain scale, i.e., blocks of size M_0, 'short' wires i.e., those within the block are assumed to have a probability of failure which is linear in their length, and contribute to the block's failure. However 'long' wires that interconnect

[1] For example the critical path in an adder becomes important only for rare inputs resulting in carries having to be propagated across the entire word.

blocks of size M_0 and above, end up being too long and unreliable, i.e., reliability becomes wire dominated. Thus it makes sense to make long wires reliable. This can be achieved by dividing a wire into shorter sections and applying redundancy to these sections, making 'long' wires wider and/or introducing a wider spacing among wires to reduce coupling. In either case making long wires reliable comes at some additional area overhead which in my assumption is modeled by the factor k_o. Relative to the model in the previous section this increases the design scaling factor d by a factor k_o in Eq 2.2. For example in the results below I let $k_0 = 3$ corresponding to the use of 3-way redundancy in sections associated with 'long' wires.

Assumption 2.3.1.b is similar to Assumption 2.2.5.b in that one may apply spatial redundancy to blocks of any size. Specifically, if n-way redundancy is applied across blocks of size M_0 gates the system would have M_S/M_0 such blocks. Then for $n = 3, 5, 7 \ldots$ the probability of failure of an n-way redundant block of size M_0 is approximated by

$$\binom{n}{\frac{n+1}{2}} (M_0 p_g + L_0 p_w)^{\frac{n+1}{2}}.$$

Then using Assumption 2.3.1.a and taking into account that 'long' wires according to the Assumption 2.3.1.c are sufficiently reliable, the probability of failure for the overall system P_S composed of $\frac{M_S}{M_0}$ such blocks is $P_S = \frac{M_S}{M_0} \times \binom{n}{\frac{n+1}{2}} (M_0 p_g + L_0 p_w)^{\frac{n+1}{2}}$.

Ignoring voters and associated circuitry, and irrespective of the block granularity M_0 at which n-way spatial redundancy is applied the overall number of gates in the system increases by a factor of n similar to the basic case considered earlier. However if replication occurs at lower levels of the hierarchy, longer wires will be required at higher levels of the design hierarchy. Moreover in contrast to the case considered previously 'long' wires occupy more area because of wire redundancy used to make them reliable. So the total area overhead of realizing n-way spatial redundancy will grow faster as it is realized at a lower level of the design hierarchy.

To properly capture the overheads when n-way spatial redundancy is applied starting at a hierarchical level M_0 I modify Eq. 1 to reflect them. For $M \leq M_0$ it

remains the same which by Eq. 2 gives an area $A(M_0)$ for a block of size M_0. For $M > M_0$ this is modified as follows. The initial condition becomes $M = M_0$. The initial area with n-way redundancy at scale M_0 is $nA(M_0)$. The differential growth in area for a system with n-way redundancy and $M > M_0$ is now given by

$$dA = \frac{A}{M}dM \ + \ k_0 n k_l (k_r \sqrt{A})(k_w(1-r)M^{r-1}dM. \tag{2.4}$$

This can be viewed as multiplying the design scaling factor by $(k_0 n)$ to capture the additional overhead associated with redundant wires. Together these considerations allow me to evaluate area-reliability tradeoffs when both gates and wires may fail – the result is summarized in the following proposition.

Proposition 2.3.2. ***(General area-reliability tradeoff)*** *Under Assumptions 2.1-2.3 and 3.15 the total system area A_S and system reliability P_S for a system with complexity M_S when n-way redundancy is applied at scale M_0 of the design hierarchy are given by:*

$$\begin{aligned} A_S(M_0) &= n a_g \frac{(\sqrt{M_0} + tdM_0^r)^2}{(\sqrt{M_0} + k_0 ntdM_0^r)^2}(\sqrt{M_S} + k_0 ntdM_S^r)^2 \\ P_S(M_0) &= \frac{M_S}{M_0} \times \binom{n}{\frac{n+1}{2}} (M_0 p_g + L_0 p_w)^{\frac{n+1}{2}}. \end{aligned}$$

As in the basic case considered earlier, one can eliminate M_0 to evaluate the tradeoff between system area A_S and system reliability P_S in certain range corresponding to M_0 in range from 1 up to M_S.

The graph in Fig. 2.6 exhibits the area/gate overhead and system reliability under this more general model where redundancy is applied at different hierarchical levels M_0 and I have fixed $p_g = 10^{-14}$ and $p_w = 0.03 p_g$.. For contrast, basic model (previously shown in Fig. 2.3) is also included in the graph showing the dramatic impact that unreliable wires have on overheads and system reliability. The graph in Fig. 2.7 exhibits the new maximum possible probability of failure per gate that can be afforded as minimal device size gets smaller with respect to minimal size of reliable a device. The graph in Fig. 2.8 shows only the bounding curves, but

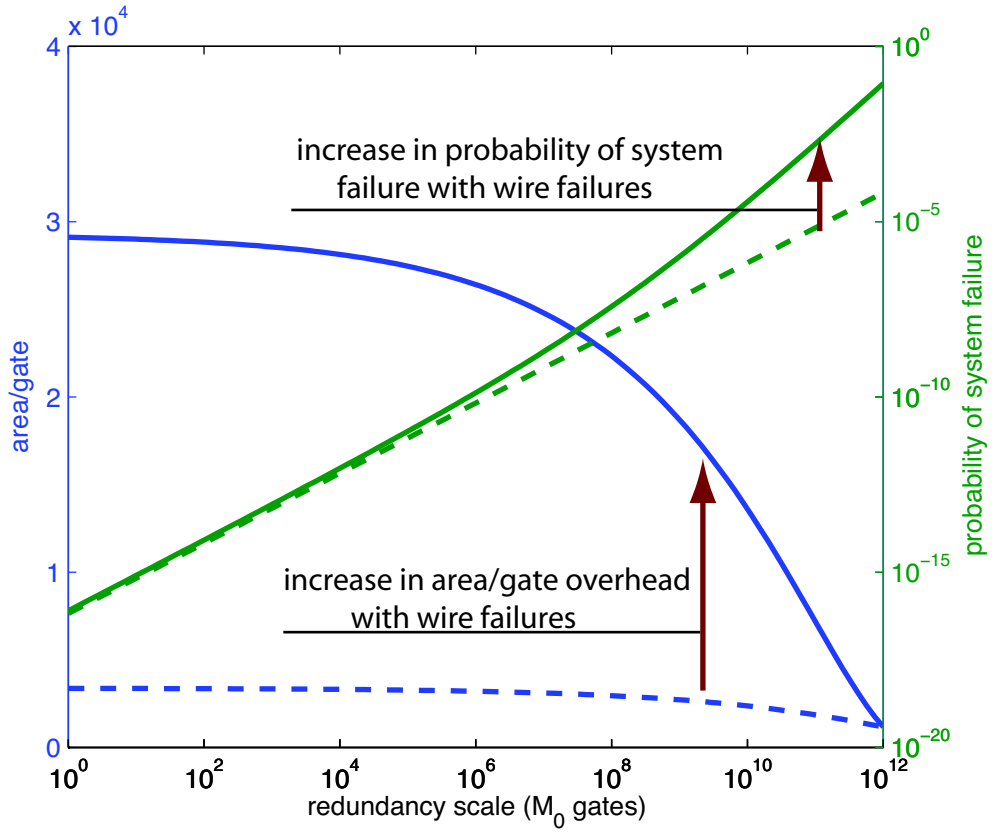

Figure 2.6: System overhead in area/gate and reliability as the hierarchical level which redundancy is applied varies.

it does this for several ratios p_w/p_g. These are akin to the results Fig. 2.5 for the basic model. Finally the graph in Fig. 2.9 shows maximum affordable probability of failure for gates (p_g) and wires (p_w) for various device scales. As can be seen in the figure the knee of the curves moves to the right as we increase density (the total range of the ratio p_w/p_g is fixed to $10^{-6} - 10^2$). This means that for higher densities the reliability of wires becomes more important than that of gates. This could be expected as at higher densities the system is increasingly in a wire dominated regime, so wire reliability is increasingly a concern.

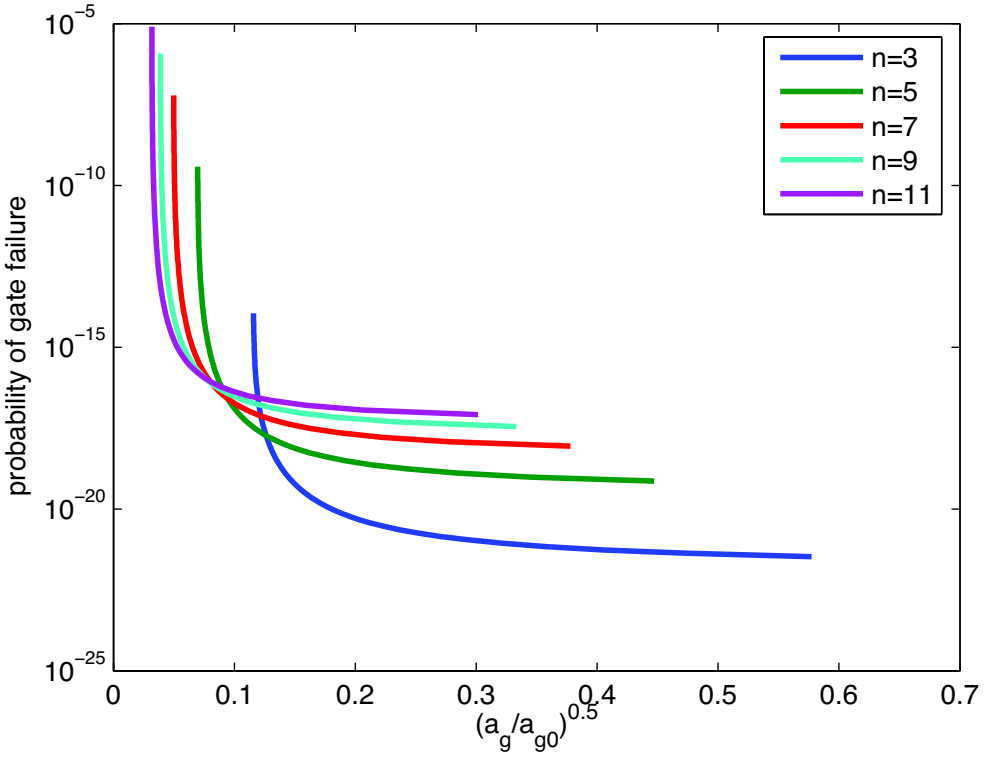

Figure 2.7: Minimum affordable reliability for reduced device size.

2.4 Defect vs Fault Tolerance

In this section I briefly discuss two points. The first is that the above wire scaling results could also be applied to also understand the overheads associated with achieving defect tolerance through brute force spatial redundancy, and the second, that in some (narrow) cases it may be interesting to consider jointly addressing defect- and fault-tolerance.

In many practical cases defect tolerance can be achieved with low overheads. The basic idea is to build the system hierarchically based on a pool of interchangeable, possibly configurable, units. When this is possible, then it suffices to test the units to determine which are defective units, and implement the desired function using the pool of operational units. For such an approach to be practical it is critical that redundancy, testing and configuration be carried out at a reasonable level of

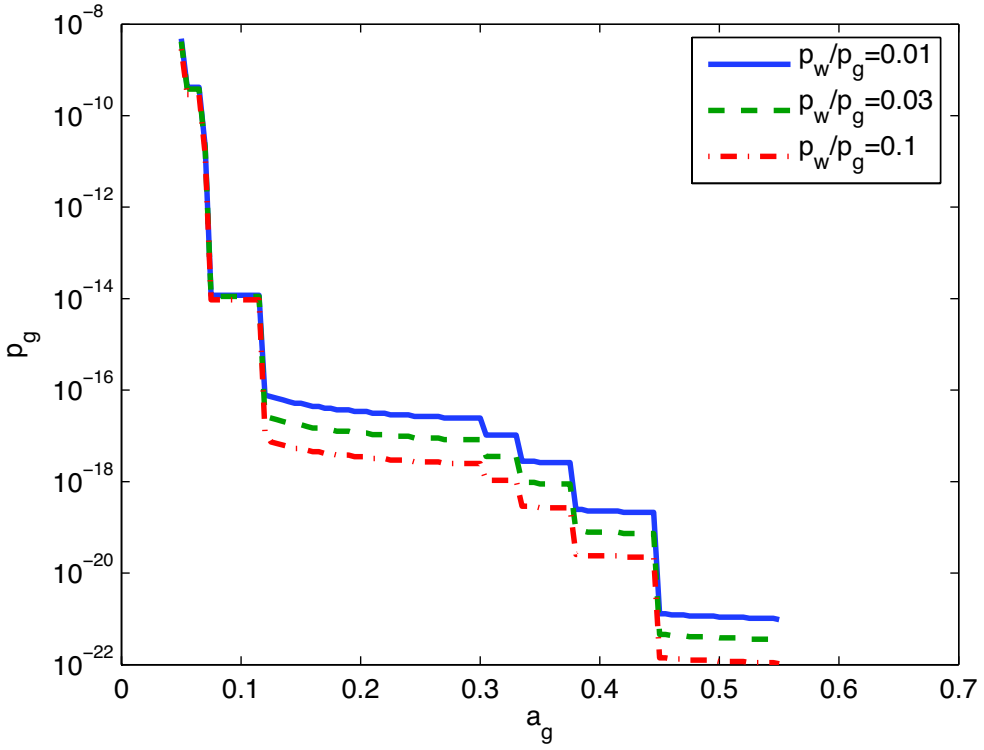

Figure 2.8: Minimum affordable reliability for reduced device size (outline).

the hierarchy. Indeed, on one hand, if this is done at a very fine grain, e.g., at the gate level, the overheads associated with mapping defects and configuration around such defects can be very high. On the other hand, applying redundancy at the chip level may be excessively costly, particularly if the defect density is such that the likelihood of a chip without any inherent defect tolerance mechanisms is unlikely to work. Thus, in practice, it is desirable to consider doing this at an intermediate level of the hierarchy where the required redundancy is fairly low, and the testing and configuration overheads are reasonable, and likelihood of failure of a unit is fairly small and only a few redundant units are needed – see e.g., [21].

The above discussion makes sense when one considers regular systems, or designs which naturally allow interchangeability of spare units. However in general irregular high performance functions may not be decomposable based on low overhead interchangeable spare units. Indeed one can envisage to use a regular pool of

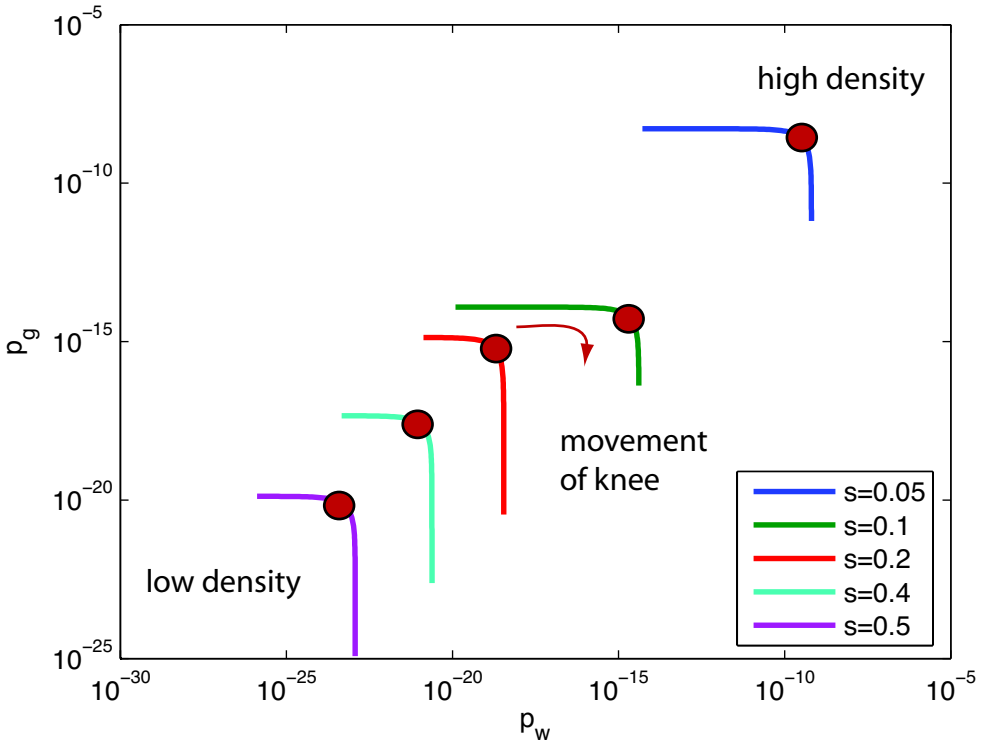

Figure 2.9: Tradeoff between gate reliability (p_g) and wires reliability (p_w) at different reduced device sizes $s = (a_g/a_{g0})^{0.5}$.

complex units, e.g., FPGAs, to implement very diverse functionality, but this would result in significant overheads. In this case the situation in terms of area overheads becomes similar to my fault tolerance model, except that redundancy orders of 2 are viable, while for fault tolerance a minimum of at least 3 and only odd numbers are considered. Also, in contrast to the case of fault tolerance where very high reliability, e.g., $1 - 10^{-16}$ is required, the target system yield, i.e., probability a system is defective, can be much lower, e.g., 99%, with some chips being discarded. Below I shall further consider this difficult case of an irregular system where defect tolerance is to be achieved through spatial redundancy.

Defect and fault tolerance techniques have a non-trivial interaction with each other when simultaneously applied. For example, if one were to apply redundancy to achieve defect-tolerance at higher level of design hierarchy than that for fault

tolerance, then the latter may mask some defects preventing testing from identifying defective blocks. In other words blocks used to achieve defect-tolerance may not fail permanently, i.e, unbeknownst to the tester they may be more susceptible to soft faults, because there degree of redundancy will be reduced by permanent defects. One would thus expect it to achieve defect tolerance below fault tolerance in the design hierarchy. But what if such masking were used to the designers advantage. Specifically when the fault rate is high and defect density low, could redundancy associated with fault tolerance also used efficiently to achieve defect tolerance. For example if a large number of redundant blocks were required to achieve the required fault tolerance. Then defects in a small number of these blocks, would still allow the system to be operational but with a slightly reduced fault tolerance which may be acceptable. Such an approach might enhance yield, while eliminating the need for sophisticated testing required for defect tolerance with reduced reliability and possibly reduced overall overheads with considering fault and defect tolerance as discrete problems.

2.5 Implications on Power Consumption

Static power consumption is primarily dependent on the operating regime of the devices and therefore is independent on wire scaling. In contrast dynamic power consumption is tightly related to wire scaling. It is well known that dynamic power dissipation is proportional to load capacitance (as well as switching frequency and square of switching voltage). Load capacitance consists of two components: the wire and the device. In the case of large devices the second component used to be dominant. However as devices scale down their capacitance decreases rapidly. Therefore in prospective technologies considered in this thesis load capacitance is likely dominated by capacitance of wires. In turn the capacitance of a single wire is proportional to its area. So, the total dynamic power consumption is proportional to total area of the wires in the system. One can see from Fig. 2.4 that the area/gate ratio is typically much greater than 1 in wires dominated case. Therefore the total

area is almost equal to total area of wires and can be used as a proxy for dynamic power consumption.

2.6 Implications for Future Technology

Traditionally prospective technologies have been associated with diminishing device size. Indeed, in the recent past, this has been a good place to invest since smaller devices provided faster and more power efficient operation per device in addition to denser integration. However this approach is reaching a point of diminishing returns. Not only are fundamental limits of atomic size not far away, but the high complexity and cost of lithography with decreased feature size makes this impractical. Moreover the benefits of smaller devices in terms of speed and power efficiency are lower as the role of wires becomes dominant. Additionally, as mentioned earlier a higher rate of faults and density of defects is expected for small devices. As I have discussed in this chapter addressing these problems would require the use of extra resources like area or power budget that can potentially render useless benefits of smaller devices size.

An alternative approach could be to increase computational power by using a large number of bigger, more reliable devices. For example, large numbers of devices appear to be the only way to implement large memory capacity. In contrast to memory, computational elements can be shared in time if they operate at high speed. However high speed operation is costly in terms of power. So it may be desirable to run more computational elements at slower speeds. One can expect a quadratic reduction in the energy per computation with a linear slowdown of computation (e.g. with reduction of voltage or by using an adiabatic switching mode if the voltage cannot be reduced). So, given a fixed power constraint, one could, for example, increase computation capacity by a factor of 10 by using a quadratic number, i.e., 100, low power devices operating 10 times slower.

Using current technology such large numbers of devices would necessitate

many chips, which is costly and requires large system. So three dimensional ($3D$) packing of devices would be desirable. Unfortunately current manufacturing technology based on photo lithography is intrinsically two dimensional. Real $3D$ technology would require new manufacturing principles (e.g. self-assembly). A more realistic approach might be to use $2D$ technology to achieve $3D$ packing of devices, e.g., one could print circuits on a thin film that can be wound around for compact filling of $3D$ volume. Thus a $2D$ system with a large area could still be placed in a compact format. This approach would allow one to pack a large number of devices in a $3D$ volume as might be required for a large memory and/or computational capacity under power constraint, but one where connectivity is still $2D$. (One could imagine interlayer wired or wireless communication, but such possibility is limited due to precise alignment problem.)

The scaling laws developed in this work are relevant in thinking about such systems. They will very likely be wire dominated due to necessity of long range communication. Circuit printing on a thin film will be technologically challenging task and feature size will likely be not as small as on a chip surface. However in light of the results in this thesis I can argue that, for the scenario discussed above, it may be more important to achieve small feature size for wires than for devices.

2.7 Conclusion

In this chapter I developed a new model for area scaling in wire dominated systems to study density-reliability tradeoffs for future technologies. The motivation was to evaluate when smaller less reliable devices make sense and at what hierarchical levels (granularity) one should incorporate spatial redundancy. To my knowledge this is the first attempt to evaluate such tradeoffs. Perhaps the most interesting result emerging from this work is a study of the tension between the reliability of devices vs wires vs density. My results indicate that wire reliability becomes more critical as the technology density increases. Although area can be used as a crude

proxy for power, it would be interesting to further enhance the model to capture power density issues.

Chapter 3

Perturbation Based Computing as a Possible Alternative Computational Model

3.1 Introduction

Advances in the synthesis and self-assembly of nanoelectronic devices suggest that the ability to manufacture dense nanofabrics is on the near horizon [11, 12, 17, 22, 25, 26, 48]. 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 [5, 11, 22]. 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 [1, 5, 23, 34]. 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 call 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 chapter, I investigate a promising new class of non-Turing computational models, called perturbation-based (also known as Reservoir Computing), and show its potential to synergistically address the two sides of the complex system design equation: technology and applications. My 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 I 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-based computing, I propose a novel hybrid eNano-CMOS platform for realizing such machines – as will be seen, the platform relies on a new style of configuration that, I believe, can directly leverage the strengths and circumvent the limitations of technologies characterized by high density but also high structural and performance uncertainty. I 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, I substantially enhanced a publically available simulation tool [33], so that it could support the large variety of machine configurations relevant to my 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 machine organizations, exposing multiple/differentiated core dynamics – see Sections 3.2 and 3.4 for details.

Note that the class of computational models investigated in this chapter was recently discovered, independently, by two research groups [27, 33] and later a similar approach was proposed by yet another group [40]. Yet, their work was driven by

research pursuits and objectives quite different from those in this thesis. Section 3.5 gives details on such prior work and establishes the uniqueness and novelty of my thesis's contributions.

Organization. The remainder of this chapter is organized as follows. Section 3.2 provides background on perturbation-based computational models. Section 3.3 introduces my proposed hybrid eNano-CMOS configurable platform and makes the case for its use for next generation IT. Section 3.4 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 3.5 contrasts the work presented in this thesis to relevant previous contributions, and Section 3.6 concludes with a discussion on future work and challenges.

3.2 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 I 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[6]. 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 [6], the fading memory[1] requirement means that "the influence of the input a long time before the given moment gradually fades out."

Mass [33], 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 [33]. 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 [33]. Given an input $u \in \mathbb{R}^{\mathbb{R}^n}$, I 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

[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 [6].

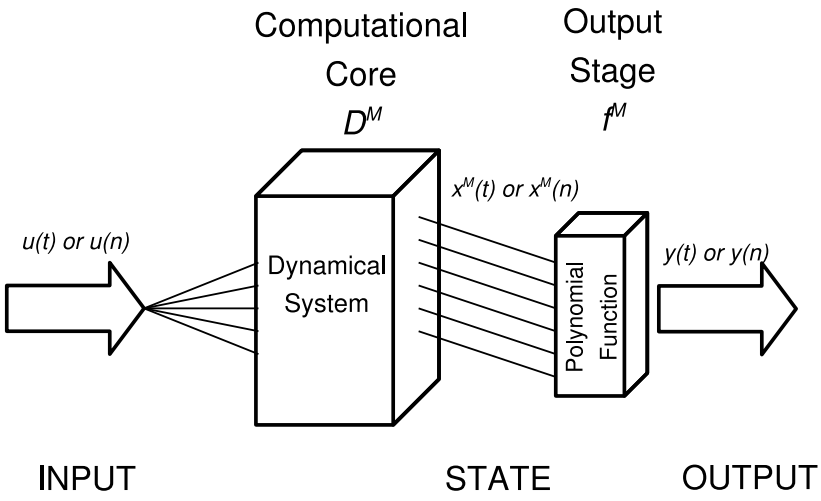

Figure 3.1: A Perturbation-based machine.

fairly simple, in that the approximation can be realized by a simple nonlinear moving average operator [6].

Perturbation-based machines. Fig. 3.1 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 [33]. 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. 3.1, 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, Mass 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 [33]. 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 I 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.

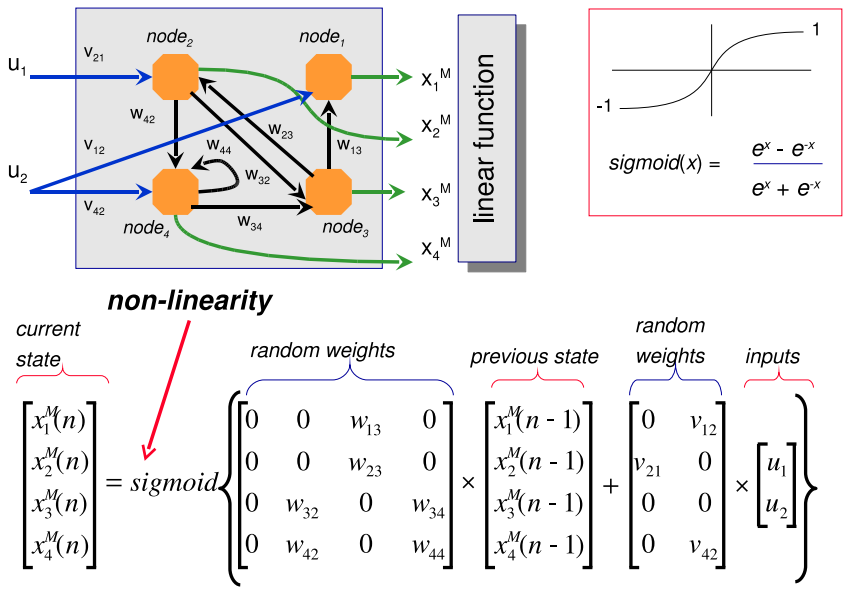

Figure 3.2: Computational core and output stage of a small discrete-time perturbation-based machine.

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 [27, 33]. 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 chapter, only linear readout maps were considered. As such, I have used standard linear techniques to determine appropriate output functions: linear regression for tasks with real-valued outputs, and linear classification for discrete outputs [28].

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. 3.2 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

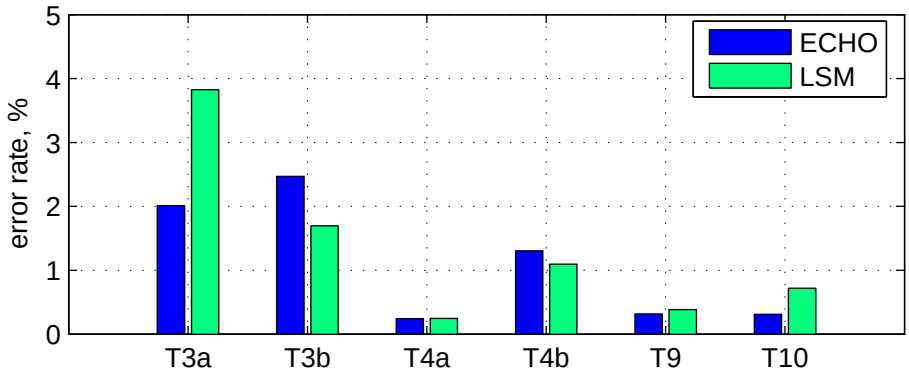

Figure 3.3: Average task error rates for a pool of machines with randomly generated computational cores, operating under ECHO-like and LSM models.

the range $[-1, 1]$) to a weighted sum of their inputs. Due to space limitations, I depict a very simple computational core comprised of a recurrent network with only 4 nodes. (For the actual experiments reported later in the chapter, much larger sparse incidence matrices defining the connections and weights of the corresponding recurrent networks were randomly generated.)[2] As shown in Fig. 3.2, 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 chapter were generated by simulating machines operating under the discrete time model illustrated in Fig. 3.2. This model is very similar to the ECHO model proposed in [27], except that in my case: (1) node-to-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

[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 [28].

[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 [33], relying on the resulting Euclidian distances

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. I also experimented with another radically different network and node model, based on leaky-integrate-and-fire nodes operating in continuous time, as in [33] – I refer to this as the LSM model. The performance results in Fig. 3.3 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, I systematically obtained equivalently good results, supporting my 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 chapter will be based on my ECHO-like model [27], which I deem to be broadly representative.

3.3 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, my intent is not to design dynamical systems to realize specific base operators for a given task. Instead, I 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 Mass, Jaeger, and others, including ourselves, shows

between core nodes, I randomly choose connections between them, using a probability low favoring shorter/local connections.

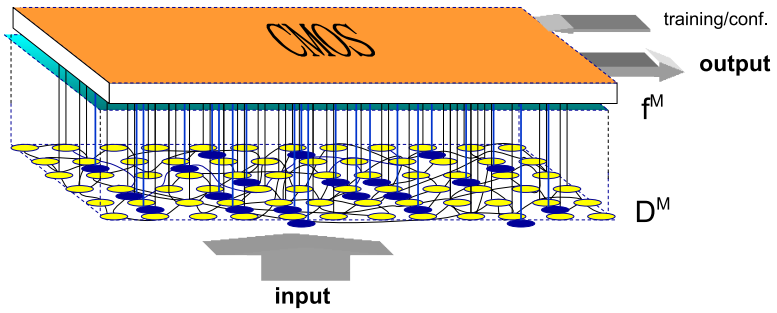

Figure 3.4: Hybrid eNano-CMOS configurable platform for perturbation-based computing.

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 thesis.

Hybrid platform. Given the previous configuration paradigm, I 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. 3.4 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. 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. 3.5 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.

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

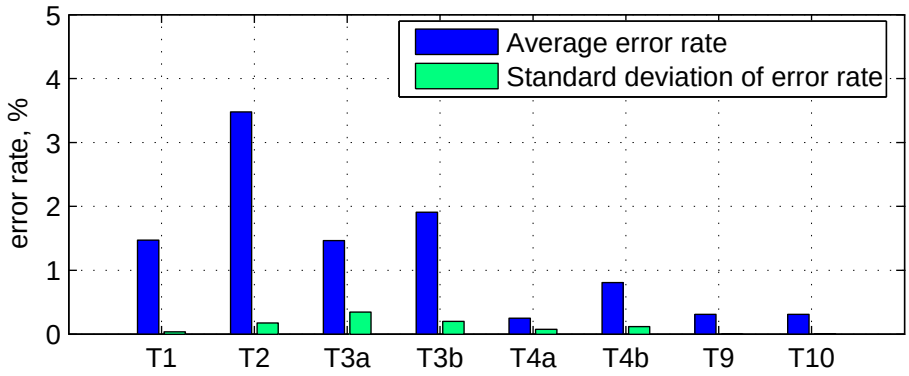

Figure 3.5: Average and standard deviation of task error rates obtained for a pool of machines with randomly generated computational cores of a target size.

This 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 my 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 noise* impacting the 'nominal' response of each element defined upon fabrication.[5] Accordingly, I assessed machine performance (task error rates) in the presence of structural noise resulting from dynamic performance variability in the computational core. Fig. 3.6 exhibits results for the following three noise regimes (assuming identical core sizes in all cases):

Scenario 1: no noise, serves as my 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. 3.6, even for the highest noise level, the error rate increases are in most cases fairly small for identical core sizes, supporting my claim that this computational model operates well under this type of persistent performance variability. Furthermore, on the bottom in Fig. 3.6 I 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 – I refer to this as 'unstructured redundancy.' This should

[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.

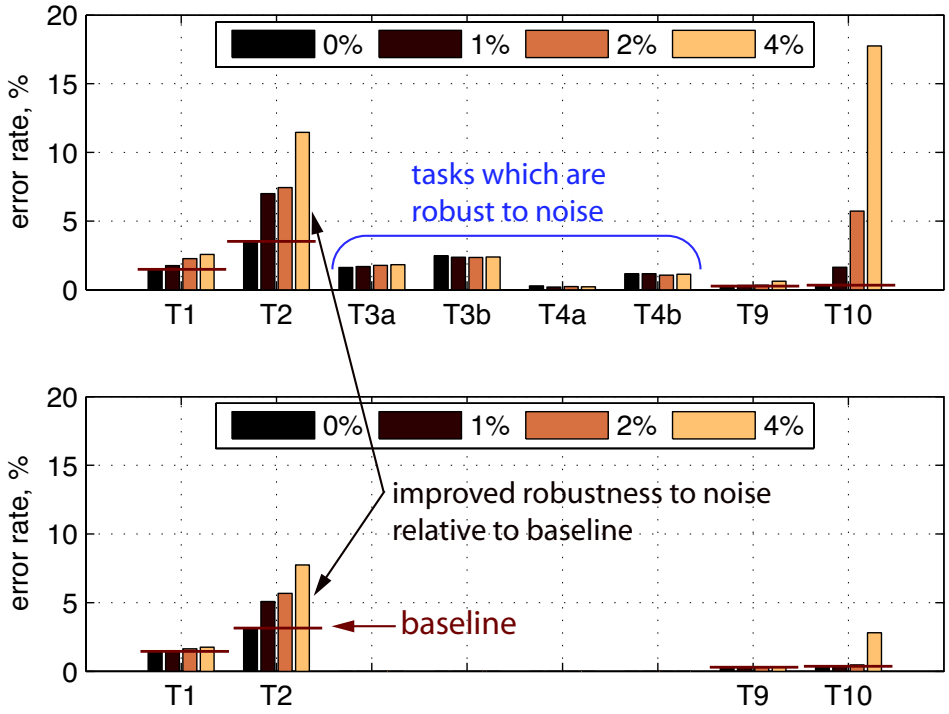

Figure 3.6: Average task error rates for a pool of machines with randomly generated computational cores, operating under three noise regimes.

be contrasted with the design complexity associated with the structured redundancy required by machines operating under traditional computational models.

3.4 A Machine Architecture for Perturbation-Based Computing

So far my discussion of perturbation-based machines assumes that they would contain a single monolithic computational core, yet in what follows I 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. 3.7.

Proposed multi-core machine architecture. A shown in Fig. 3.7, I envisage an architecture that has multiple computational cores (i.e., reservoirs of

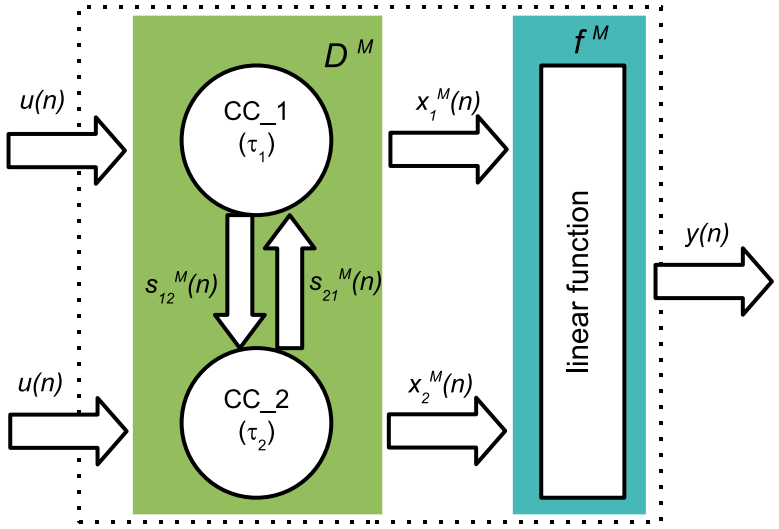

Figure 3.7: Multi-core machine architecture.

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, I 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. 3.8 (denoted by an m in the figure's legend) illustrate this – as can be seen, machine performance for a set

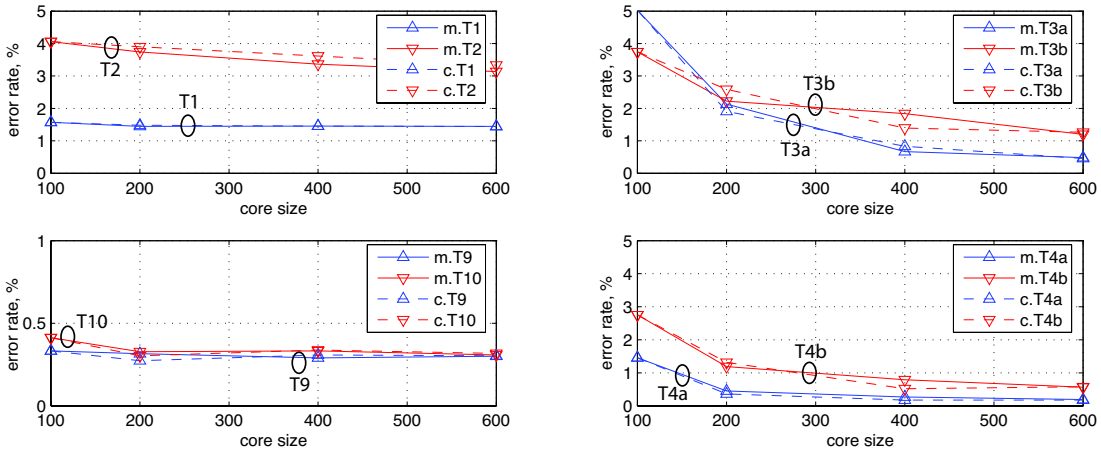

Figure 3.8: Test error rates for benchmark tasks, considering monolithic core machines ('m') with increasing core sizes, and multi-core machines ('c') with the same combined core sizes.

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. 3.8 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. 3.8, 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.

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

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

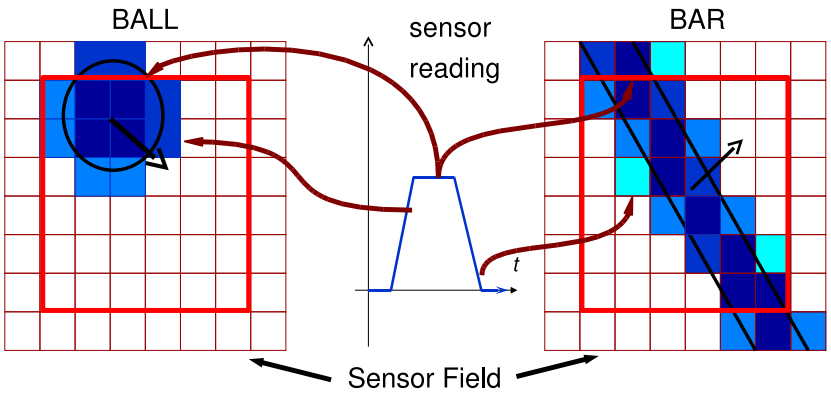

Figure 3.9: Vision task: predicting object movement on an $8x8$ array of sensors.

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. 3.9, where I use color intensity to represent the percentage of each sensor field currently covered by the moving object [46]. 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. I 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) [46]. 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 $2 \times 2 \times 3$ 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, my 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 [9]. 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

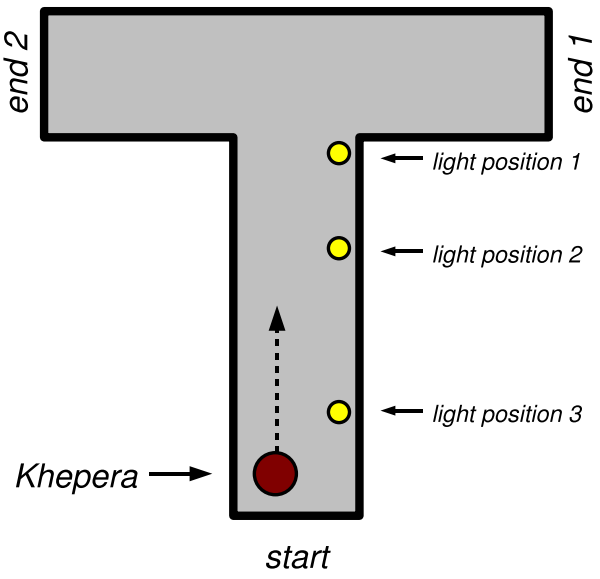

Figure 3.10: Three robot navigation tasks on T-Maze: Task 5 (light at position 1), Task 6 (light at position 2), and Task 7 (light at position 3).

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.

Below, I present results for an experiment illustrating the scalability enhancements one can achieve via this unique type of decomposition. I consider the classical T-maze robot navigation task – as shown in Fig. 3.10, the task is to have a robot (Khepera [41]) 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. 3.10, 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. Such robot control schemes are referred to as a reactive control according to [2].

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. 3.5, for example, I 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 my 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 my 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 – I can provide more insightful experimental results for this class of machines.

Accordingly, given a machine configuration, target task, and desired performance, in the sequel I 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. I 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).

I first assessed the performance of *monolithic core* perturbation-based machines, for the three T-maze tasks. For Task 5, I 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, I found that a smaller machine configuration (comprising only a 50 node core), was still able to deliver my top performance ('less than 2% error rate'), yet with a decreased yield of 50% (my 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%. My experiments also showed that this same small configuration (with a single 50 node core), was able to deliver my 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.

I now present and discuss the results obtained for Task 6 – with the light placed at some intermediate point of the corridor (see Fig. 3.10). In contrast to Task 5, much larger core sizes were consistently required to achieve similar performance and yield targets. For example, to deliver my lower target performance ('less than 10% error rate') with my 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 my 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 I am 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 I 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. 3.10), I was not able to achieve any of my 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 I 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. 3.11 summarizes the results delivered by such decomposition for Task 7 (the 'harder' task). The top graph of Fig. 3.11 gives the combined core size of several machine configurations capable of delivering my top performance point (error rate under 2%) with a yield of 50% and 75%. Namely, for each slowdown factor

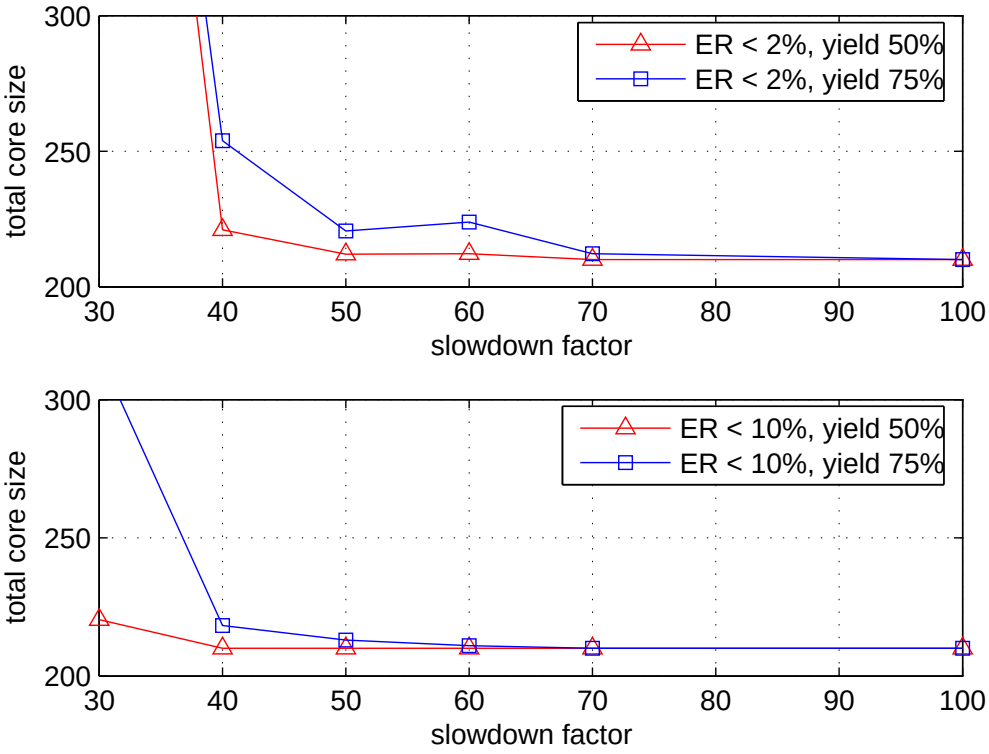

Figure 3.11: Size of double core machines for T-maze task T7, considering different relative core speeds, and different target performances and yields. On all experiments, the size of the fast core is 200 nodes. The top and bottom figures show the combined core sizes required by machine configurations exhibiting a target error rate (ER) of less than 2% and 10%, respectively, as the slowdown factor of the slower core increases.

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 my lower target 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. 3.11 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 my 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, I 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 my 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 my 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.

3.5 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 the general the 'rest' state; 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 [28]. 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. Perturbation-based computing was independently proposed by two research groups [27, 33], both of which have furthered this area, but have fundamentally different objectives than

ours. The main objective in [33] 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 [33, 45, 46]. These issues are not germane to my research, and in fact obscure my core objectives. The work of [27] 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 [28]. 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, my 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, I am 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.

3.6 Conclusions and Future Work

I proposed a hybrid eNano-CMOS platform for realizing perturbation-based machines, relying on a new style of configuration that, I believe, can potentially leverage the strengths of emerging nanoelectronic technologies. In particular, I 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, I then proposed a multicore machine architecture and novel design and decomposition principles, exploiting task specific contextual and temporal scales. I experimentally

demonstrated the effectiveness and promise of these, for a set of benchmark tasks.

Given the promising results reported in this chapter, 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 chapter. 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 [37], some promising adaptive directions are explored towards this end, for a class of abstract perturbation-based machines which I denote standard ECHO model [27]. Still, in my 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. 3.4.

I 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.

3.7 Appendix

The performance numbers reported in this work 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 work, except for those associated to the experiments reported in Figs. 3.3, 3.5, 3.6. In those, for each benchmark task I used the smallest core size(s) leading to performance saturation. For example, as can be seen in Fig. 3.8, 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. 3.6 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. 3.6) 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. 3.6 use three times that density – once again, aiming at reducing the impact of structural noise on performance.

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

Domain 1: Wireless Communications. I 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 [10]. 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 [29].
- **Task 2**: Channel equalization - using a modified (highly nonlinear) channel model. I substantially amplified the nonlinear distortion of the original model in [29], in order to increase the difficulty level of the task.

Domain 2: Voice. I 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 [46]. 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[32]. Depending on the task, the output identifies the word or the gender of the speaker. I 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 [3]. 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 [41] 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. 3.10. 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 [46]. 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 [46].

- **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.

Chapter 4

Conclusions and Future Research

4.1 Summary of Key Results and Contributions

In this thesis we discussed trends in microelectronics advancement and associated problems in design for future systems with high computational capacity. The first direction was a scaling analysis in the wire dominated case. The main contributions here are the:

- development and analysis of an area scaling model in wire dominated regime;
- and the use of this scaling model to analyze the overheads associated with achieving fault-tolerance by applying spatial redundancy at different levels of the system hierarchy.

The second direction was an exploration of perturbation-based computational models as a promising choice for implementing next generation embedded systems using future technologies. Contributions here are:

- proposal and validation of design principles for perturbation-based machines;
- and the proposal of a hybrid eNano-CMOS platform for realizing perturbation-based machine, that can potentially make efficient use of future technologies.

The following perturbation-based computation design principles are proposed and validated in this thesis:

Design Principle 1: Defect-tolerance, randomly assembled cores and configuration/training. Our first principle is that one need only ensure

that the core and readout connectivity are sufficiently 'rich' to achieve the desired approximation after training. In other words the designer need only control the size and statistics (e.g., connectivity) of the core network without precisely specifying its topology. Manufacturing defects and heterogeneity in the non-linearities within the network thus become part of its intrinsic randomness of the network. The experiments shown in Fig. 3.5 empirically support this point. These results show the task error rates achieved, across a suite of benchmark tasks, T1-T10 realized on randomly generated computational cores of a similar (sufficiently large) size. As can be seen there is negligible variation in their ability to perform the task, i.e., have essentially the same computational power – as assessed based on task error rates.

Design Principle 2: Fault-tolerance through unstructured redundancy. Our second principle is that 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-module-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 a reduction in design cost in comparison to structured redundancy. A disadvantage is that this comes at a cost, bigger cores will consume more power.

Design Principle 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.

Design Principle 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 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.

Design Principle 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 to perform careful core design, but simply define some large scale characteristics for connectivity and dynamics of its constituent subnetworks.

While there has been an increasingly active community of researchers exploring the potential of the perturbation computing (also known as reservoir computing) paradigm for real time processing tasks, little has been reported on possible implementation and design principles. This thesis serves to set down some of the possible design principles that would support the "design" of such systems if they were based on randomly assembled networks.

4.2 Future Work

Advancements in microelectronics over the last few decades have been enabled by progress in photolithography technology providing increasingly smaller feature size. Smaller feature size means denser integration (more transistors per chip) as well as faster and more energy efficient transistors. The nanoelectronics era has followed this major research and development effort attempting to carry technology forward along similar lines. Nanoelectronics promised great benefits 10–15 years ago, when feature size was hundreds nanometers. However nowadays, with feature size already basically at the nano scale (35nm–45nm), prospective benefits become questionable. The shrinking of feature size is becoming increasingly costly. Problems associated with small device size also become more pronounced, and ultimately the fundamental limit of device size is not as far away as it was 10–15 years ago (the silicon crystal lattice spacing is 0.54nm).

Considering all these, the only feasible way to achieve higher computational capacity is to use a significantly larger number of devices. Since now we cannot reduce device sizes to make them more energy efficient; each device has to operate in a slow low-power mode to meet the overall system power dissipation limit. This means that the number of devices has to grow rapidly (quadratically) with target growth in computational capacity, while the cycle time (or equivalent timing characteristic) has to become longer.

If we try to put an increased number of devices on a chip, while device size stays the same, this would require an enormous chip area. Indeed traditional photolithography places devices inside a thin 2D layer on the chip surface. In order to keep system size reasonable it is desired for new technologies to pack this huge number of devices in 3D volume. True 3D integration is problematic through photolithography (creating many thousands of layers, doing this layer by layer would be expensive). Self assembly is a possible way to achieve 3D integration. But, so far, there are no such technologies, so it could be a promising direction for future

work. A possible alternative could try be reuse available photolithography by printing circuits on a large area thin flexible film. Later this film could be compacted into a small 3D volume. Such thin film technology may be not so unrealistic and therefore presents an interesting direction for future work.

It may be hard to construct a specific desired structure using self assembly (either to achieve 3D integration, or to achieve deep nanoscale or both). In that sense perturbation-based computing is an interesting example of the use of random structures for computation. It is reasonable to assume that such random structures are easier to construct through self assembly. Therefore another prospective research direction is how to build, and how to employ for computation, random structures in 3D and/or at deep nanoscale.

As mentioned above, we expect future high computational capacity systems will consist of very large numbers of devices operating at relatively slow speeds. Corresponding future research directions include analysis of scaling laws for such systems (e.g. presented in this thesis area scaling in wire dominated case) and suitable computer architectures for such big systems.

Bibliography

[1] SEMATECH. International Technology Roadmap for Semiconductors - 2007 edition on emerging research devices. http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[2] Eric A. Antonelo, Benjamin Schrauwen, and Dirk Stroobandt. Event detection and localization for small mobile robots using reservoir computing. *Neural Networks*, 21(6):862–871, 2008.

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

[4] D. Bhaduri and S.K. Shukla. Reliability evaluation of von Neumann multiplexing based defect-tolerant majority circuits. *Nanotechnology, 2004. 4th IEEE Conference on*, pages 599–601, 16-19 Aug. 2004.

[5] G. Bourianoff. The future of nanocomputing. *Computer Magazine*, pages 44–49, Aug. 2003.

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

[7] Andrew E. Caldwell, Yu Cao, Andrew B. Kahng, Farinaz Koushanfar, Hua Lu, Igor L. Markov, Michael Oliver, Dirk Stroobandt, and Dennis Sylvester. GTX: the MARCO GSRC technology extrapolation system. In *DAC '00: Proceedings of the 37th Annual Design Automation Conference*, pages 693–698. ACM, 2000.

[8] Philip Christie and Dirk Stroobandt. The interpretation and application of Rent's rule. *IEEE Trans. Very Large Scale Integr. Syst.*, 8(6):639–648, 2000.

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

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

[11] A. DeHon. Array-based architecture for FET-based nanoscale electronics. *IEEE Trans. Nanotechnology*, 2(1):23–32, 2003.

[12] A. DeHon, S. G. Goldstein, P. J. Kuekes, and P. Lincolnn. Non-photolithographic nanoscale memory density prospects. *IEEE Trans. Nanotechnology*, 4(2):215–228, 2005.

[13] J.S. Denker. A review of adiabatic computing. In *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, pages 94–97, Oct 1994.

[14] W. Donath. Placement and average interconnection lengths of computer logic. In *IEEE Transactions on Circuits and Systems*, volume 26, pages 272–277, 1979.

[15] William Evans and Nicholas Pippenger. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, 44:1299–1305, 1995.

[16] Richard Phillips Feynman. *Feynman Lectures on Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

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

[18] Bruce E. Hajek and Timothy Weller. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, 37(2):388–391, 1991.

[19] Jie Han and Pieter Jonker. A defect- and fault-tolerant architecture for nanocomputers. *Nanotechnology*, 14(2):224–230, 2003.

[20] Guoqiang Hang. Adiabatic cmos gate and adiabatic circuit design for low-power applications. In *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, volume 2, pages 803–808, Jan. 2005.

[21] C. He, M. Jacome, and G. de Veciana. A reconfiguration-based defect tolerant design paradigm for nanotechnologies. *IEEE Design & Test of Computers - Special Issue on Advanced Technology and Reliable Design of Nanotechnology Systems*, 22(4):316–326, July-August 2005.

[22] J. Heath. Wires, switches, and wiring: A route toward a chemically assembled electronic nanocomputer. *Pure and Appl. Chem.*, 72(11), 2000.

[23] J. R. Heath. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280:1716–21, June 1998.

[24] Ron Ho, Ken Mai, Hema Kapadia, and Mark Horowitz. Interconnect scaling implications for CAD. In *ICCAD '99: Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 425–429, 1999.

[25] Y. Huang, X. Duan, Y. Cui, K.H. Kim L. J. Lauhon, and C. Lieber. Logic gates and computation from assembled nanowire building blocks. *Science*, 294(5545):1313–1317, 2001.

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

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

[28] 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.

[29] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 304(2):78–80, April 2004.

[30] Philip J Kuekes, Warren Robinett, Gadiel Seroussi, and R Stanley Williams. Defect-tolerant interconnect to nanoelectronic circuits: internally redundant demultiplexers based on error-correcting codes. *Nanotechnology*, 16(6):869–882, 2005.

[31] B.S. Landman and R.L. Russo. On a pin versus block relationship for partitions of logic graphs. *Computers, IEEE Transactions on*, C-20(12):1469–1479, Dec. 1971.

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

[33] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

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

[35] Gordon E. Moore. Cramming more components onto integrated circuits. pages 56–59, 2000.

[36] K Nikolic, A Sadek, and M Forshaw. Fault-tolerant techniques for nanocomputers. *Nanotechnology*, 13(3):357–362, 2002.

[37] Mustafa C. Ozturk, Dongming Xu, and Jos C. Prncipe. Analysis and design of echo state networks. *Neural Computation*, 19:111–138, 2006.

[38] G. Roelke, R. Baldwin, and D. Bulutoglu. Analytical models for the performance of von neumann multiplexing. *Nanotechnology, IEEE Transactions on*, 6(1):75–89, Jan. 2007.

[39] Daniel P. Siewiorek and Robert S. Swarz. *Reliable computer systems (3rd ed.): design and evaluation*. A. K. Peters, Ltd., Natick, MA, USA, 1998.

[40] Jochen J. Steil. Backpropagation-decorrelation: online recurrent learning with O(N) complexity. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, volume 1, pages 843–848, Jul 2004.

[41] Theodor Storm. KiKS is a Khepera simulator. http://www.tstorm.se/projects/kiks/.

[42] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. *Automata Studies*, pages 43–98, 1956.

[43] Michael Wehner, Leonid Oliker, and John Shalf. Towards ultra-high resolution models of climate and weather. *International Journal of High Performance Computing Applications*, 22(2):149–165, 2008.

[44] Michael Wehner, Leonid Oliker, and John Shalf. A real cloud computer. *IEEE Spectrum*, (10):24–29, October 2009.

[45] W.Maass and H.Markram. Theory of the computational function of microcircuit dynamics. In S.Grillner and A.M. Graybiel, editors, *The Interface between Neurons and Global Brain Function*, volume 93 of *Dahlem Workshop Report*. MIT Press, 2005.

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

[47] Yuan Xie, Gabriel H. Loh, Bryan Black, and Kerry Bernstein. Design space exploration for 3D architectures. *J. Emerg. Technol. Comput. Syst.*, 2(2):65–103, 2006.

[48] V. Zhirnov and D. Herr. New frontiers: Self-assembly and nanoelectronics. *IEEE Computer*, 34(1):34–43, January 2001.

Index

Vita

Andrey Zykov was born in Moscow, Russia on November 3rd, 1976, the son of Vyacheslav Mihailovich and Tatiana Vasilievna Zykovy. He received the Bachelor of Science and Master of Science degrees from the Physical and Quantum Electronics Department of Moscow Institute of Physics and Technology, Russia in June 1999. After working as software engineer in different IT companies in Moscow, he joined the Electrical and Computer Engineering Department of the University of Texas at Austin in January 2003, where he was awarded with Microelectronics and Computer Development (MCD) Fellowship. During the summer of 2008 and the fall of 2008 he interned at Magma Design Automation, Austin, TX. He is a student member of IEEE, ACM and Sigma Xi.

Permanent address: Krasnostudencheskiy pr-d, d.2, kv.182
127434, Moscow, Russia

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.