The Challenges and Opportunities of Nanoelectronics

Seth Copen Goldstein
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
412-268-3828 / seth@cs.cmu.edu

Abstract

Nanoelectronics presents the opportunity of incorporating billions of devices into a single system. Its opportunity is also its challenge: the economic design, verification, manufacturing, and testing of billion component systems. In this presentation I will explore how the abstractions used in computer systems change as we approach nanoscale dimensions.

1. INTRODUCTION

Future computing systems will inevitably be built using nanoelectronics, i.e., from devices and wires with feature sizes below thirty nanometers. The SIA roadmap [15] predicts that traditional silicon-based systems will have feature sizes of below 40nm within the decade. There are also advances being made in building computing systems using new technologies, such as molecular electronics [2]. Successfully harnessing nanoelectronics requires a rethinking of the abstractions and models that are the basis of designing computing systems. While each technology has its own unique requirements, we show that new abstractions are necessary strictly because the feature sizes are nanoscale.

The ever increasing improvement in processor performance is fueled by the ever increasing number of available faster transistors, but it is driven by a hierarchy of abstractions. One plausible hierarchy is: Transistors $\rightarrow$Logical Gates $\rightarrow$Circuits $\rightarrow$Blocks $\rightarrow$ISA $\rightarrow$Programs. Each abstraction layer hides the details of the layer below it; helping to control the complexity of designing and implementing a systems with hundreds of millions of components. They also promote a separation of responsibilities; allowing independent progress to be made at different levels of the system. In this paper we examine how current trends in semiconductor manufacturing as well as how new technologies, e.g., molecular electronics and self-assembly will influence the hierarchy of abstractions used to build computing systems.

From the circuit designers perspective the hierarchy begins with transistors and works it way through logical gates to functional blocks. Computer architects use the functional blocks created by circuit designers to create an instruction set architecture (ISA). The ISA is the abstraction used to define the processor to programmers. The architect implements the ISA by combining control, storage, and datapaths. Most of today's processors are organized as von Nuemann computers: they have a single point of control (one program counter) which controls a deeply pipelined datapath which has many function units. Intermediate values are stored in a register file while a single main memory is used to store programs and data. The success of this model is that it is easy to think about and program. However, we need to rethink this model if we are to efficiently harness the vast number of devices, limit the effect of increased wire delay, control design complexity, decrease the power required, and control the cost of manufacturing.

2. FABRICATION

Perhaps the greatest impact of the nanoscale on electronics will be the reduced ability to arbitrarily determine the placement of the components of a system. The most extreme example of this is to be found in chemically assembled electronic nanotechnology (CAEN), a form of molecular electronics which uses bottom-up assembly to construct electronic circuits out of nanometer-scale devices. Large-scale molecular electronics requires some form of self-assembly [2]. When using self-assembly, individual devices and wires are first manufactured, and only later assembled into a circuit. While self-assembly promises to be a very economical process (compared with the cost of traditional semiconductor fabrication), it cannot be used to create the arbitrary patterns that can be formed using photolithography. Only simple, crystal-like structures, can be created using self-assembly. Furthermore, defect densities of self-assembled circuits are projected to be orders of magnitude higher then in silicon-based devices. Thus, self-assembled circuits and architectures will have to be designed for defect tolerance.

To a lesser extent these same problems will appear as traditional semiconductor technology continues to scale. The complexity of generating a reliable mask set which produces reliable chips is already limiting the ability to create arbitrary patterns of wires. This can be seen in the trend towards "structured" ASICs, which allow custom chips to share many of the same masks [18]. As devices scale down it is also harder to maintain constant characteristics for all the devices on a single chip [11]. Some argue that process variation will essentially eliminate the performance gains typically expected when feature sizes shrink [1]. These trends indicate that as feature sizes shrink even photolithographically manufactured chips will need to be crystal-like, i.e., built from very regular structures.

In order to implement useful reliable functionality on top of crystal-like structures, post-fabrication customization is required; this customization will be used for two purposes (1) to implement the desired functionality and (2) to eliminate the deleterious effects of the defects [4, 6]. Economics will speed the movement towards customizable chips. As mask sets become more expensive it becomes more economical to reuse the same chip for different tasks, i.e., to use programmable hardware (also called a reconfigurable fabric) such as field programmable gate arrays (FPGAs). A reconfigurable fabric is a network of processing elements connected by a programmable interconnect [7]. It can be programmed by determining how signals are routed on the interconnect. The desire to decrease time-to-market is also accelerating the trend towards using FPGAs for ever higher volume applications.

Defects in manufacturing have been, until now, primarily the concern of process engineers, not circuit designers or architects. In the era of nanoelectronics, delivering chips which can be viewed as defect free will likely be too expensive. In fact, this is happening already; for example, state-of-the-art FPGA chips with known defects can be purchased at a discount [19]. The defective chips can be used because the defects on the particular chip are determined not to affect the customer's design. In the future, defect tolerance will have to be designed in at the circuit and architectural level. One method is to use reconfigurable fabrics. Reconfiguration provides defect tolerance by configuring the desired circuit around the defects, thus creating a reliable system from an unreliable substrate. Before the fabric is shipped its defects are mapped [9]. When the chip is used, the desired circuit is configured around the defects. The main challenge here is to develop architectures and tools which can—in the field—quickly place-and-route (P&R) circuits around the defects. Final P&R needs to be done in the field so that a single configuration can be shipped for all devices, in spite of the fact that each device will have a different set of defects.

3. DEVICES

Nanoelectronic devices, and here we also include wires, pose their own challenges. The transistor has thus far proven an extremely useful building block for digital logic. In addition to its switching properties, it can be used to construct logical gates which isolate their inputs from their outputs (I/O isolation) and it has gain which promotes noise immunity [8]. Some nanoelectronic technologies have no equivalent device. In addition, as dimensions scale down, the wires used to connect the devices become the dominant source of signal delay and power consumption [15].

Isolation and gain, while not a *sine qua non* for constructing large scale digital systems, is required if one is going to build a large system from smaller subunits ones without having to redesign the subunits. We should point out that isolation and gain are not strictly a property of the device, but usually arises from the device and the design methodology being employed. For example, static CMOS is a design methodology which uses the transistor to construct logical blocks. In static CMOS, the inputs to a logical block are always applied to the gate of the transistor, which is isolated from its other two terminals. Furthermore, a small change on the gate can cause a large change on the output due to the connections to power and ground. There are other transistor-based design methodologies, such as complementary pass logic (CPL) [13], in which the inputs and outputs of a logical block are not isolated. Furthermore, there is signal loss—instead of power gain—and, there is no way to invert a signal, so whenever a function is computed its complement must also be computed.[1]. Large scale CPL circuits require periodic buffers which restore signals and provide isolation; allowing individual subsystems to be composed together to form larger systems.

Some of the proposed logic families for non-silicon nanoelectronics are similar to CPL in that gain and isolation are not intrinsic in every circuit element. QCAs are an example of such a system. Designers of QCA based circuits include a system clock which helps to isolate inputs from outputs as well as add gain to the system [12]. A more traditional approach is found in nanoFabrics [6] which proposes to use diode-resister logic combined with clocked molecular latches for gain and isolation. What these systems (CPL, clocked-QCA, and NanoFabrics) share in common is they provide logical switching separately from isolation and restoration. In order to manage this extra design dimension we propose a new model, SirM. SirM stands for **s**witching, **i**solation, **r**estoration, and **M**emory. We call a set of devices a complete SirM family if it contains devices which support all the components of SirM, and therefore, can be used to construct a scalable complete logic family.

An example of a complete SirM family in the domain of molecular computing is:

- **Switching** is based on diode-resistor logic using resistors and molecular diodes. As with CPL, functions and their complements must be constructed since it is impossible to construct an inverter out of just diodes and resistors.
- **Isolation** comes from a molecular latch combined with a clocking methodology as described in [14].
- **Restoration** occurs through the use of the above latch.
- **Memory** can be constructed either directly from molecules (e.g., the rotaxane [10]) or from the latch.

SirM is an example of the easiest kind of change to the abstraction hierarchy. It can be thought of as a new layer between the new physical devices and today's abstraction of a transistor. In other words, a "transistor" like contract can be established by combining the proper switch, isolator, and restorer devices all together. Therefore, we are able to maintain the current abstraction, providing backward compatibility

[1]If the complements of all the inputs are available it is always possible, by Demorgan's Law, to compute the function and its complement.

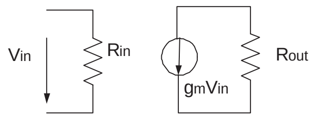

Figure 1: **The Norton equivalent circuit for one circuit block**

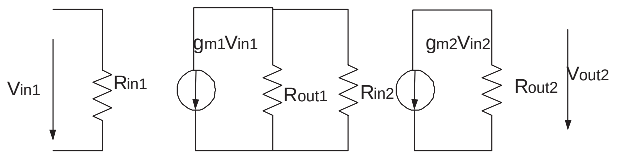

Figure 2: **The equivalent circuit for cascaded circuit blocks**

for tools and ideas while providing the opportunity to pierce the transistor abstraction when necessary. This provides the design community with an incremental path towards harnessing the underlying devices directly. Furthermore, it does not change any of the contracts with other layers in the hierarchy.

However, if isolation is separate from switches—which is currently the case for all two-terminal device families—then some local analog design is necessary when constructing networks of switches. Any two-terminal voltage controlled circuit to be modeled from its input and output impedances and its gain. For example, Figure 1, shows a model of a two-terminal device. The gain of this device is $g_{\mathrm{m}}R_{\mathrm{out}}$. If two of these devices are placed in series, as in Figure 2, then the gain of the total circuit is $\frac{V_{\mathrm{out2}}}{V_{\mathrm{in1}}} = g_{\mathrm{m1}}\frac{R_{\mathrm{out1}}R_{\mathrm{in2}}}{R_{\mathrm{out1}}+R_{\mathrm{in2}}}g_{\mathrm{m2}}R_{\mathrm{out2}} = g_1\frac{1}{1+R_{\mathrm{out1}}/R_{\mathrm{in2}}}g_2$. We can see that if the input, e.g., R_{in2}, does not have infinite input impedances, i.e., it does not provide isolation, then the gain of a device depends on the structure of the circuit. The variation in an individual device's characteristics is even more pronounced if the circuit has fan-in or fan-out. In fact, every possible path in a circuit needs to be analyzed in order to determine when a restorer should be inserted into the circuit. CPL, when combined with static CMOS buffers (or inverters) is an example of a complete SirM family which requires such analysis [20].

All the methods of isolation that we are aware of include either transistors or a clocking strategy. An important challenge for the nanotechnologists in the molecular computing community is to develop a design methodology and an associated two-terminal device which isolates its inputs from its outputs without requiring a clock. Additionally, whatever strategy is employed should not require global analysis in order to understand the behavior of a local structure. In other words, the input to a gate should have a very large impedance, e.g., R_{in2} in Figure 2 should be as close to infinity as possible.

4. PROCESSORS

Computer architects have improved processor performance by increasing the use of parallelism and decreasing the overhead of long communication times. This has been done while maintaining the abstraction of a single thread of control, i.e., programs are specified in a sequential language. These improvements have been enabled by technology scaling: smaller devices run at faster speeds and increase the total amount of resources available. The internal structure of a modern processor, is designed to exploit as much parallelism as possible within the sequential stream of instructions being executed. All of the improvements have come at the cost of increased power consumption and increased complexity. Furthermore, technology trends (e.g., longer wire delays, faster clocks, and, increased power densities) are already limiting the continued application of current techniques. Nanoelectronics will exacerbate all of these trends as well as increase the rate of transient faults.

The level of integration available with nanoelectronics suggests a radically different approach: Implement programs directly in hardware. This approach forces one to abandon two important abstractions: the instruction set architecture (ISA) and sequential control. The ISA is the primary abstraction used by computer architects to hide the internal structure of the processor. It defines a fixed set of instructions which determine how the processor is used by the programmer (or a compiler). The main advantage of an ISA is that it allows a programmer to write down a sequential list of instructions to define a program. With the advent of sophisticated tools, such as optimizing compilers, the necessity for writing down such a low level description of a program becomes less important. More importantly, the ISA—almost by definition— impedes the ability to fully exploit the microarchitecture.

We suggest that compiling directly to the hardware will open up opportunities for creating custom compute engines tailored directly to the application of interest. The compiler will create a configuration for a reconfigurable fabric. This model, often called spatial computing, is more suited for the regime of nanoelectronics since it directly supports many levels of parallelism. It can essentially create a custom datapath for the application under execution. Of course, it would be nearly impossible to program by hand, and relies heavily on compiler technology and CAD tools.

Tailoring the hardware directly to the program, e.g., spatial computing, has the potential to overcome the negative effects of scaling. By eliminating the ISA and allowing tools, such as compilers, to manipulate the underlying hardware structures directly one can optimize not only for time, but also for other important metrics in the nanoelectronics design space, e.g., defect/fault tolerance or power. Let us look at lowering power dissipation. Dynamic power (P) is a function of capacitance (C), voltage (V), the activity factor (α), and switching frequency (F), $P = \frac{1}{2}CV^2\alpha F$. One way to reduce power is to reduce α, the number of devices that switch per cycle by using asynchronous circuits [16]. Asynchronous circuits eliminate the global clock (and its associated global wire), and are based on local communication and synchronization. In sharp contrast to synchronous methodologies, devices in asynchronous circuits only switch when they are actually computing. Another orthogonal approach is to decrease clock frequency by exploiting parallelism. A fundamental result of early VLSI research is that for many functions there is tradeoff between the implementation area (A) and the time it takes to compute the function (T): $\frac{1}{A} \propto T^n$, where n is between 1 and 2 [17]. If we fix the time a computation takes, then since total computation time is inversely proportional to clock frequency (F) we obtain, $F \propto A^{-1/n}$, and, $P \propto CV^2A^{-1/n}$. In scaled CMOS, Further power saving are saved because the maximum switching speed is related to voltage, $F \propto (V - V_{\text{th}})^{5/4}/V$ [3], or in the the relevant range if we fix V_{th}, $F \propto V$ [5] and $P \propto CF^3$. Thus, $P \propto CA^{-3/n}$ and since capacitance is roughly proportional to area, $P \propto A^{-2/n2}$. If we can effectively harness the area we can achieve the same performance while reducing clock frequency and power.

5. CONCLUSIONS

Nanoelectronics holds the promise of continuing technology scaling to feature sizes below 40nm. However, to harness the abundance of resources and new constraints concomitant with this feature size requires a new approach to the abstractions used to construct computing systems. Instead of clean barriers which hide the details between layers of the design, we need to develop abstractions which allow tools to manage the details. Instead of using a hardware description language to design circuits for a fixed architecture on a chip assumed to be defect free we suggest using a high-level programming language which can be compiled into configurations which are then mapped onto a reconfigurable fabric avoiding any defects that might be on a particular chip.

Acknowledgments

This research is funded in part by the NSF under Grants CCR-9876248 and CCR-0205523 and by DARPA under contracts MDA972-01-03-0005 and N000140110659.

References

[1] Keith A. Bowman and James D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE JSSC*, 37(2), Feb. 2002.

[2] M. Butts, A. DeHon, and S. Goldstein. Molecular electronics: Devices, systems and tools for gigagate, gigabit chips. In *ICCAD-2002*, Nov. 2002.

[3] K. Chen, C. Hu, P. Fang, Ren Lin, and D.L. Wollesen. Predicting cmos speed with gate oxide and voltage scaling and interconnect loading effects. *IEEE Trans. Electron Devices*, 44(11):1951–57, Nov 1997.

[4] C. P. Collier, E. W. Wong, M. Belohradský, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath. Electronically configurable molecular-based logic gates. *Science*, 285:391–394, July 16 1999.

[5] Michael J. Flynn, Patrick Hung, and Kevin W. Rudd. Deep-submicron microprocessor design issues. *IEEE Micro*, July 1999.

[6] S.C. Goldstein and M. Budiu. NanoFabrics: Spatial computing using molecular electronics. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 178–189, June 2001.

[7] Reiner Hartenstein. A decade of research on reconfigurable architectures - a visionary retrospective. In *Proc. International Conference on Design Automation and Testing in Europe 2001 (DATE 2001)*, Exhibit and Congress Center, Munich, Germany, March 2001.

[8] Robert W. Keyes. What makes a good computer device? *Science*, 230(4722):138–144, October 1985.

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

[10] N. Spencer M.V. Martinez-Diaz and J.F. Stoddart. The self-assembly of a switchable [2]rotaxane. *Ang. Chem. Intl. Ed. Eng.*, 36:1904, 1997.

[11] Sanil Nassif. Delay variability: Sources, impacts and trends. In *Proceedings of the ISSCC*, pages 368–9, 2000.

[12] Michael Niemier and Peter Kogge. Exploring and exploiting wire-level pipelining in emerging technologies. In *Proceedings of the 28th International Symposium on Computer Architecture*, 2001.

[13] J.M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, 1996.

[14] D. Rosewater and S.C. Goldstein. Digital logic using molecular electronics. In *IEEE International Solid-State Circuits Conference*, February 2002.

[15] Sematech. *International Technology Roadmap for Semiconductors*, 2001.

[16] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989. The 1988 Turing Award Lecture.

[17] J.D. Ullman. *Computational Aspects of VLSI*. Computer Science Press, 1984.

[18] Ron Wilson. 'structured' asics arrive. *EETimes*, May 5 2003.

[19] Inc. Xilinx. Virtex-ii series easypath: Frequently asked questions. "http://www.xilinx.com/publications/products/v2/faq/faq101_easypath.pdf", January 2003.

[20] H. Zhou and A. Aziz. Buffer minimization in pass transistor logic. In *Proceedings of the International Symposium on Physical Design (ISPD-00)*, pages 105–110, N.Y., April 9–12 2000. ACM Press.

[2]For scaled CMOS, static power is becoming an important consideration and so one could both balance an increase V_{th} and a decrease V to reduce dynamic and static power at the cost of using more hardware.