TERAMAC: POINTING THE WAY TO REAL-WORLD NANOTECHNOLOGY

David Clark

Shortly after the millennium you could be using a computer made with trillions of nanometer-sized components that were manufactured in a test tube, rather than with chips made of silicon. How soon this happens will depend on two factors. The first is physics: complementary metal-oxide semiconductor (CMOS) technology will run into physical limitations around the year 2010, according to most researchers. The second—and probably more important—is economics: Gordon Moore's so-called Second Law says the cost of the fabrication facilities used to build chips is increasing faster than the growth in demand for chips. Fabrication costs are projected to be $30 billion to $50 billion by 2012—a substantial fraction of the entire market for chips. "It's very likely that the economic consequences of Moore's Second Law could be the factor that causes his first law to end," says Stan Williams, who leads the Basic Research Program at HP Labs in Palo Alto. (Roughly, Moore's first law states that microprocessor capacity doubles every 18 months.)

In what must be a cosmic coincidence, Williams and other researchers say that the ability to grow molecular-sized electronic components—already being produced in research labs around the world—is expected to reach commercial economic and technical feasibility around the same time CMOS technology reaches its physical or economic limit.

However, before computers are emblazoned with stickers that say, "Dupont (or Monsanto) Inside," some thorny problems must be solved. For example, components that are chemically synthesized (a process called *self-assembly*) and then chemically connected to form electrical circuits (called *self-ordering*) will inevitably have defects—possibly many. Also, a major goal of nanoscale technology is to build systems that incorporate a huge number of computational devices (researchers speak in terms of a mole, approximately 6×10^{23}). How will tomorrow's computer architects achieve the organization that allows the entire mass of devices to operate efficiently? Chemically assembled machines will certainly have to reproduce the arbitrary complexity that general-purpose computation demands.

Teramac (see Figure 1), an experimental configurable computer built at the HP Lab, has demonstrated a viable solution to this problem. The powerful computer was originally designed to test different parallel-computing architectures in the mid-'90s, but has proven that a massively defect-tolerant computer—that accommodates the uncertainty of "grown" computing devices—can be built. (For specifics on Teramac, see the sidebar.)

Working around defects

Mostly for economic reasons, HP intentionally built Teramac with an unknown percentage of defective components. After they built the machine, they configured it to test itself. After Teramac identifies its defects, it maps different logic configurations onto the remaining defect-free components. (The machine contains almost 220,000 hardware defects—any of which would cause a traditional silicon computer to fail.) Teramac's defect-tolerant design incorporates a high-communication bandwidth that facilitates mapping around the defects. This bandwidth can be configured to operate over 100 times faster than a high-end single-processor workstation, for some applications.

Of course, the idea of defect-tolerant computing is not entirely new. For example, John Von Neumann wrote a theoretical statistical-analysis paper about computing with unreliable resources in the early 1950s. Teramac, more than 40 years later, is more than theory, and is based on the work and ideas of many researchers since Von Neumann.

Lessons for nanocomputers

Although Teramac employs conventional silicon IC technology, many of the problems associated with the machine are similar to the challenges that nanoscale-computer architects face. "What's interesting here is that we have computer engineers talking to chemists about building computers. It's a proof in principle of our strategy for designing computers at the scale of a molecule," says Philip Kuekes, one of Teramac's architects at HP Labs. "The computer is built out of crossbars, basically, and a fat-tree architecture interconnect. ... The structures it's built out of are very regular. When Jim Heath (a UCLA chemist involved in the Teramac project) saw the structure, he said, 'These are just like crystal structures. ... We know how to build these.'"

And, as Kuekes points out, ang-

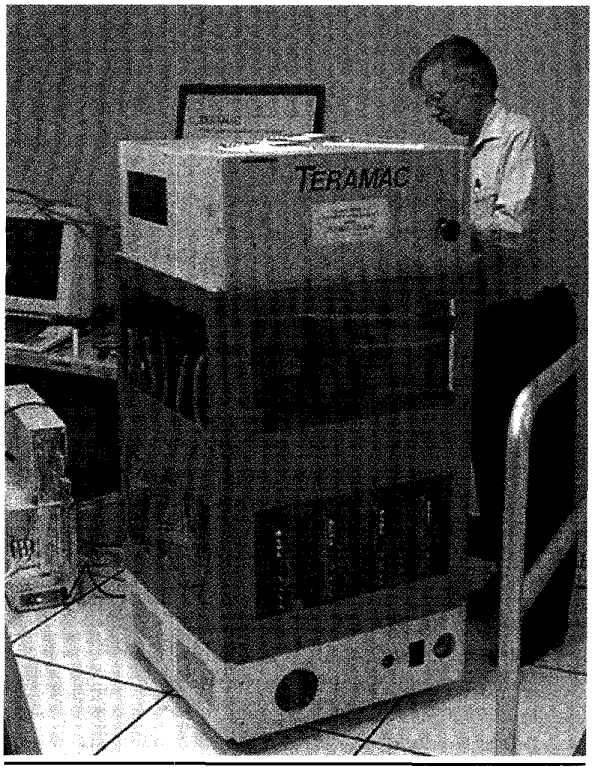

Figure 1. Teramac, with David Kuekes, one of its architects.

strom-scale devices are already being built. "It's being done all the time. Carbon nanotubes, for example—which are, in a sense, big molecules. But the problem is how to construct *electrical* devices chemically—and then, how do you wire them up? How do you connect four-bit atoms, independent of the fact that you might be able to grow a transistor smaller than the wavelength of light?" Duplicating even a simple silicon device such as a half-adder—which merely adds two bits together—in a self-assembled chemical device at the molecular level is extremely unlikely.

The key, according to Kuekes, is to have an abundance of switches, wires, and bandwidth. "It's not that the Teramac architecture is defect-tolerant, although that's important," he says. "The fact that it's ordered—and elemental—enough is what proves a similar machine could be built with nanometer-sized components, with the necessary functional complexity introduced with software after the machine is built. Because Teramac uses FPGAs and because we've wired them together in a very specific fat-tree architecture that lets the compiler deal with the whole thing, in effect the whole machine becomes one giant FPGA that a compiler can logically think about.

"The decision we made in terms of architecture," he goes on, "was to spend extra money on hardware in terms of the

Teramac Details

Teramac's name comes from *tera*—10^{12} operations per second (10^6 logic elements—or gates—operating at 10^6 Hz)—and *m*ultiple *a*rchitecture *c*omputer.

Hardware

Teramac is the largest defect-tolerant machine ever built. It contains 864 identical HP-designed FPGA chips in this configuration: eight printed circuit boards, each with 12 layers of interconnects for four *multichip modules*. Each MCM has 33 layers of wiring to interconnect a total of 27 chips—eight used for their lookup tables (LUTs) and 19 for their crossbars (interconnect wiring). Each computing element performs a six-input, one-output combinatorial logic function. Inexpensive, defect-prone ribbon cables connect the printed circuit boards.

Fat-tree/crossbar architecture

Each single parent node on the logic tree is replaced by several nodes. Communication between levels of the tree occurs through crossbars that connect multiple nodes at each level. At every junction of the crossbar is a switch. The crossbar not only provides a means of mapping many configuration bits together into some desired sequence, but also represents a highly redundant wiring network. Between any two configuration bits are a large number of pathways, which implies a high communication bandwidth within a given crossbar (represented logically as a fat tree). $2n^{1/2}$ address lines are needed to address n switches.

Memory

Teramac performs logic functions in memory—it stores truth tables in 64-bit LUTs. Each LUT holds the equivalent of 10 logic gates. The machine has 65,536 LUTs. Four Mbits (65,536 × 64 bits) of configuration memory define the logic functions of all the computing elements. Configuration memory is drawn from approximately 30% (256) of the FPGAs—the majority of FPGAs are used only for communication and signal routing. (It was significantly less expensive to design and manufacture a single FPGA and ignore the LUTs on the chips that are used only for communication).

Defects

Tests have determined that 10% of the logic cells in the FPGAs used as processors are defective and 10% of the interchip signals are unreliable. Out of a total of 7,670,000 resources in Teramac, 3% are defective.

Software/configuration

The use of FPGAs allows the loading of the desired architecture onto Teramac through an automated software routine. Teramac uses an extreme version of the very long instruction word (VLIW) architecture. This *indefinitely* (or insanely) *long instruction word* (ILIW) is essentially the translated logical description of the desired configuration. Teramac uses a 300-Mbit word. Any given configuration of Teramac uses only 70% to 90% of the healthy resources.

HP engineers have configured Teramac in a number of real-world parallel architectures for applications such as translating magnetic-resonance-imaging data into a 3D map of arteries in the human brain, and as a volume-visualization engine referred to as the Cube 4 architecture. Teramac has run as high as 10^{12} gate operations per second in one configuration.

Scaling

The compiler algorithms are dominated by the partitioning time, which scales linearly with the number of gates in the design. Experiments with various-size partitions of Teramac show that the time required to find defects also scales linearly with the total number of wires and switches in the fat-tree architecture. This means that the extension of this architecture to a mole (approximately 6×10^{23}) of devices looks promising.

Testing

Testing involves two stages: First, running configurations measure the state of the configurable computer. Second, a set of algorithms run on these measurements to determine the defect.

The actual testing is performed by downloading designs called "signature generators." These are sets of LUTs that generate long pseudorandom number strings that are sent around Teramac by a large number of different physical paths. If the bit stream is both correctly generated and transmitted by the network, all the resources used are probably (but not always) good. If any group fails, Teramac checks other configurations that use the resources in question in combination with other devices. Those resources found in the intersection of the unreliable configurations are declared bad and logged in a defect database.

There is a paradox in having a device test itself when it doesn't know if anything is working. According to HP engineers, only a small percentage of resources (used for clocks and to get data out of the system for observability) must be perfect. Also, a small percentage of the resources must work to guarantee that the defect-finding algorithms will work in self-testing. For this "privileged" set of resources, the designers deliberately added redundancy to insure a high probability of survival.

number of switches and wires above what most reconfigurable computer architects would use in a design. With that extra capacity our software problems suddenly got easier by a dramatic factor. The software is very efficient at finding defects and at compiling programs on the machine. By having 10 times as many bits for switching and routing, we compile 100 times faster. That hardware-software trade-off isn't necessarily what a commercial manufacturer should do, because their customers care about the cost of every chip." At present, the Teramac approach isn't economically competitive with conventional CMOS technology because so many of the resources in configurable computers are not used (for example, almost all the lookup tables (LUTs) in the Teramac chips).

There'll be some changes made

Nanotechnology, along with scientists developing the ability to build things bottom-up (from atoms upward), is a hot topic in today's popular scientific press. What is not usually discussed is that our existing structures and architectures will need to change—sometimes drastically—to use these new technologies efficiently. If we're really going to build computers at a molecular scale, huge parallelism will result. A sensible way of dealing with that, at least according to the HP engineers, would be to use an architecture such as Teramac. "The parallelism will be there," Kuekes says, "and it will be cheap—relatively speaking. Therefore the objective should be to use some of that parallelism to work around defects and create complex structures out of rather simple crystalline-like ones." ♦

David Clark *is a freelance writer based in Torrance, California, specializing in high-performance computing issues. Contact him at dwclark@earthlink.net.*