an Article from SCIENTIFIC AMERICAN

AUGUST, 1999 VOL. 281 NO. 2

Raw Computation

One of the main engines of the Oxygen project is the Raw microchip, which has wiring that can be automatically reprogrammed for different tasks

by Anant Agarwal

The Oxygen project is based on the premise that computation will eventually become as freely available as air. To achieve this goal, however, computer scientists, software designers and electrical engineers must rethink the basic architecture that underlies current computer systems. My colleagues and I in the Raw project at the M.I.T. Laboratory for Computer Science are developing an entirely new kind of microprocessor for the Oxygen project. Called the Raw chip, it will deliver unprecedented performance, energy efficiency and cost-effectiveness because of its flexible design: by exposing its wiring to the software system, the chip itself can be customized to suit the needs of whatever application is running on it.

The relentless miniaturization of microprocessors has paved the way for the Raw chip. In 1987 a microprocessor containing about 100,000 transistors and capable of performing 20 million instructions per second (MIPS) could fit on roughly one square centimeter (0.16 square inch) of silicon. But in 1997 a microprocessor with the same computing power could fit on a chip only one millimeter square. And in 2007 a 20-MIPS microprocessor will fit on a chip only one tenth of a millimeter square—one ten-thousandth the size of the 1987 microprocessor. We are entering an era in which each microchip will have billions of transistors. Clearly, we have an amazing opportunity before us.

We can, of course, fritter away this opportunity. One way to do so would be to continue advancing our chip architectures and technologies as just more of the same: building microprocessors that are simply more complicated versions of the kind built today. The problem is that the current architecture for microprocessors does not scale. Most personal computers use an interface called the Instruction Set Architecture, or ISA, between the hardware and the software. The instructions in the ISA move data from storage locations on the microprocessor to function units where the data are added, multiplied or otherwise processed. For example, an instruction might say: "ADD, Register 7, Memory Location 1,024, Register 8." This instruction directs the microprocessor to add the contents of Register 8 and Memory Location 1,024 and to store the resulting sum in Register 7. But most instruction sets do not tell the software where the memory locations or function units reside on the chip, so current microprocessors must use hardware—for example, sets of wires or buses—to connect every memory location with every function unit.

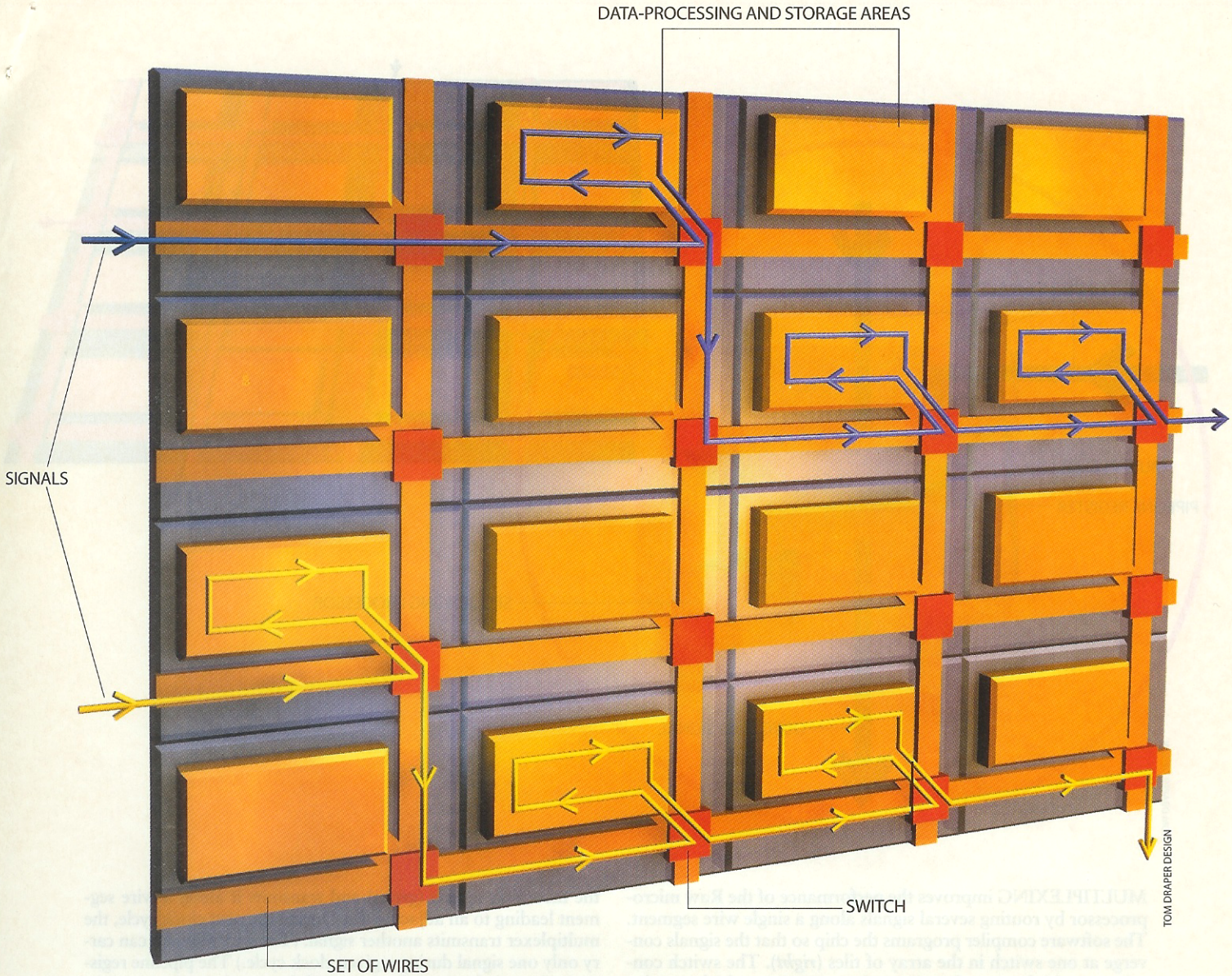
The ongoing reduction in transistor sizes will enable hardware designers to squeeze more storage locations and function units onto each chip. Smaller transistors will also lead to a decrease in the duration of the chip's clock cycle, which is the time required to perform a basic operation such as addition. But because the current architecture requires that the chip's wires connect every memory location with every function unit, the lengths of the wires will remain proportional to the diameter of the chip and will not decrease along with the clock cycle. Delays in moving data along the wires will become increasingly significant and will eventually set a limit on the chip's performance. The current architecture will also result in less energy-efficient microprocessors, because longer wires require more energy to switch signals.

Getting around the Problem

Some would argue that we have already hit a brick wall in terms of complexity, speed and energy efficiency in our existing architectures. Every personal computer has a microprocessor inside, but if you want to take full advantage of the machine you also have to buy several add-on cards, such as a modem card, a graphics card, a sound card, a math card, an FM radio card and a video card. You need to buy a big case for your personal computer simply to plug in all these specialized cards. Once the cards are installed, the system as a whole can deliver adequate performance for various multimedia applications.

But how do these special cards deliver the required performance? Some hardware designer has very carefully handcrafted the wiring on the cards' customized chips to match each specific application: video, radio and so on. The hardware experts have hand-fashioned the wires to fit the needs of the application, tailoring the circuits so that the wires are as short as possible and all the signals get from their origination point to the right place at exactly the right time. A huge amount of effort goes into this process.

So how can we get around this brick wall blocking the improvement of computer performance? We propose to solve the problem by throwing logic gates at it. A logic gate is an arrangement of transistors that controls the direction of electric current on a microchip and hence the flow of information. In about 10 years every chip will have billions of logic gates. Chip designers can take advantage of this surfeit by constructing a software compiler that uses the abundant logic gates to reroute the flow of information on the chip's wires. Instead of forcing chipmakers to spend so much time carefully laying out the wires for each application, we are going to build a processor and a compiler that permit us to



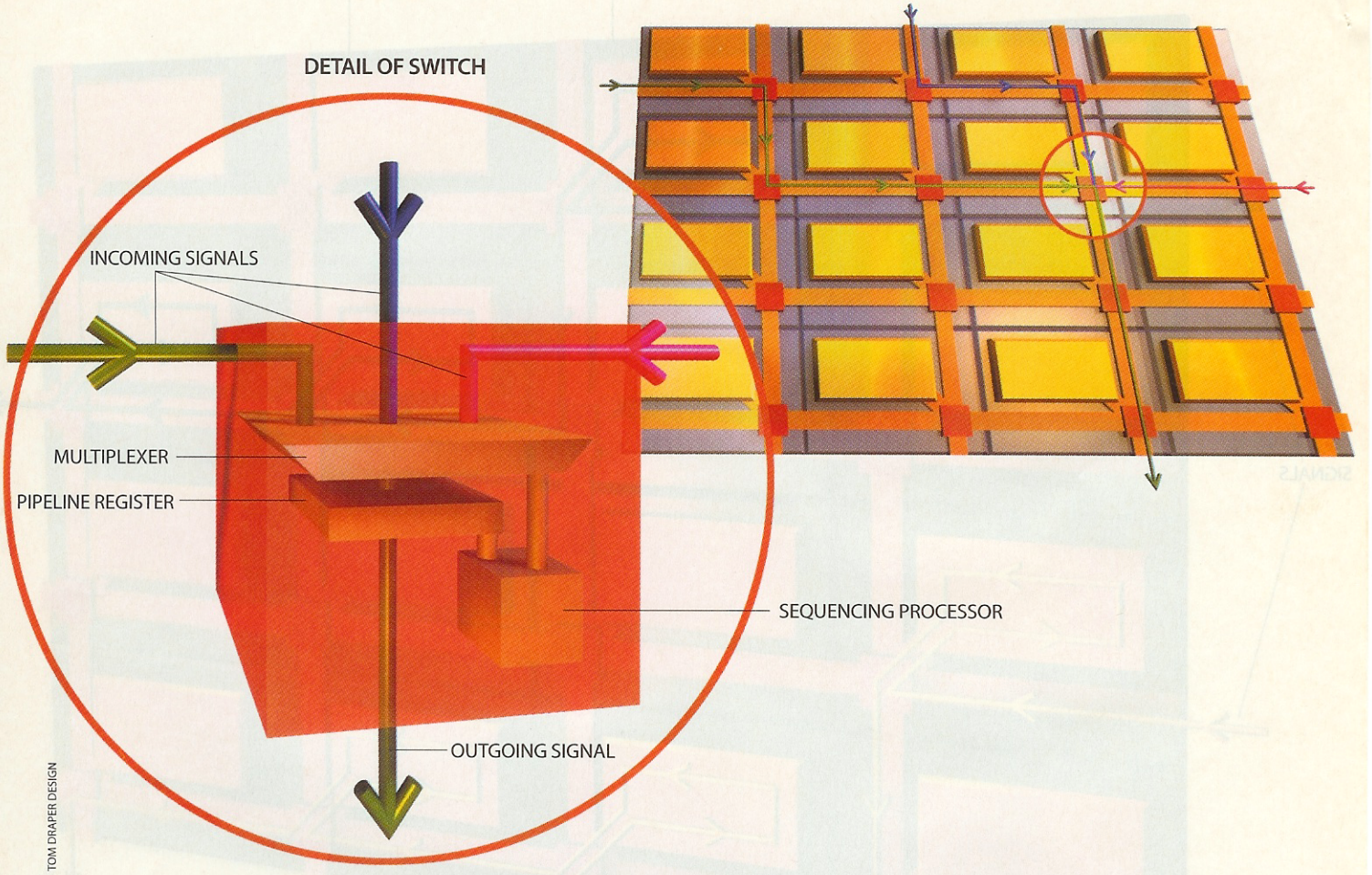
RAW MICROPROCESSOR is a rectangular array of many identical tiles. (Only a small portion of the microchip is shown above.) Each tile contains memory locations that store data and function units where the data are processed. Areas for data processing and storage are represented by the gold rectangles within the tiles. Signals flow through sets of wires that connect each tile to its neighbors. Switches at the wire junctions direct signals to data-processing areas or to adjacent tiles. The pathways of the signals are determined by a software compiler that programs the switches to meet the needs of whatever application is running on the microprocessor. The chip can run more than one application at a time; for example, it can direct a stream of video data (*blue*) along the optimal pathway for a video application, while simultaneously guiding an audio signal (*yellow*) along the path best suited for a radio application.

reconfigure the wires automatically. The software compiler will be able to take applications written in human-readable languages, such as C and Java, and map them directly into the chips.

This new model of computation is called Raw because it exposes the raw hardware on a chip—including the wires—to the software compiler. By using the free logic gates to direct and store the signals that run through the chip's wires, the compiler basically customizes the wiring for each application. This is a radical departure from existing architectures, in which software controls the chip's logic operations—basic functions such as "ADD" and "SUBTRACT"—but not the chip's wiring. In contrast, Raw allows the software to program the wires, the microchip's most valuable resource.

The layout of the Raw chip is very simple. The chip itself is an array of many tiles. Each tile is identical to all the others and contains memory units, which are collections of memory locations, and function units. More important, each tile has a switch that controls the wires connecting the tile with adjacent ones.

The excess logic gates are devoted to this switch. The compiler programs the switches on all the tiles to issue a sequence of commands that determine exactly which set of wires to connect at every cycle in the chip's operation. Thus, the compiler and the software system choreograph how data move around the entire chip by programming each of the



MULTIPLEXING improves the performance of the Raw microprocessor by routing several signals along a single wire segment. The software compiler programs the chip so that the signals converge at one switch in the array of tiles (*right*). The switch contains a multiplexer and a sequencing processor, which are also programmed by the compiler (*left*). Depending on the needs of the application running on the chip, the multiplexer selects one of the incoming signals (*green*) and transmits it along a wire segment leading to an adjacent tile. During the next clock cycle, the multiplexer transmits another signal. (The wire segment can carry only one signal during a given clock cycle.) The pipeline register in the switch stores the data until the signals can be transmitted. Multiplexing increases the carrying capacity of the chip's wires and thus reduces delays in moving data across the chip.

switching blocks. The customized signal routing effectively rewires the chip for each application.

As a first step, the compiler "pipelines" the chip's wiring so that long wires do not incur long delays. It does this by introducing registers for storing data along the wires, essentially breaking them up into multiple segments. When a wire is pipelined, a signal does not have to traverse the entire length of the wire during a clock cycle; it traverses only one segment and is then stored in a register. Because the duration of the clock cycle can be much shorter, the Raw architecture can greatly improve clock frequency, or the number of cycles that a chip can complete in a second. Raw chips will be able to achieve clock frequencies on the order of 10 to 15 gigahertz by 2010, compared with frequencies of about 500 megahertz for today's microprocessors. Although communicating a signal along a pipelined wire will take multiple clock cycles, many signal values—one for each segment—will be able to travel down the wire simultaneously. After the first signal value arrives at its destination, subsequent signal values will arrive at the end of every clock cycle, thereby increasing the signal throughput, or carrying capacity, of the wire.

The compiler also attempts to place signal values in memory locations close to the function units that will process the data. This minimizes the number of cycles that the signal values spend traveling from one location to another.

The next step comes from the realization that because the chip's wires are such a critical resource, using a wire to connect only two locations on a chip is wasteful. Rather we would like to "multiplex" each wire segment so that it can connect a large number of storage locations and function units. Multiplexing is similar to merging the feeder roads from several cities into a superhighway. Signals from the feeder connections arrive at one end of the wire segment, and a multiplexer constructed from logic gates ensures that only one signal is transmitted along the segment during a given clock cycle. The compiler programs the multiplexers to

select the appropriate signals at the right times for each application. Just as a superhighway carries more traffic than a feeder road, a multiplexed wire carries many more signals than an ordinary wire.

Finally, the compiler routes the signals along the optimal pathways by precisely scheduling the signals to meet the demands of the application. Because the chip's wires are programmed by the software, we like to call them "soft wires."

A major advantage of this design is that it can bring massive streams of data—for instance, video or sensor information—directly to the parts of the chip where computation takes place. The faster data input will yield far better performance and energy efficiency than is currently possible. The Raw chip we are building will have more than 1,000 input-output pins that can be dedicated to data streams—10 times more than the number of such pins in today's microprocessors.

By exposing its wiring to the software system, the chip itself can be customized to suit the needs of whatever application is running on it.

One Chip Fits All

The Raw chip could be incorporated into a single device that could perform a wide variety of applications: encryption or speech recognition, games or communications. We have dubbed this proposed 21st-century tool the Handy 21. A user would be able to tell the Handy 21, "Hey, turn yourself into a cell phone." The device would then locate the appropriate configuration software, download it and configure the wires of the Raw chip inside to give it the characteristics of a cell phone.

Today I carry a beeper. I also carry a cell phone and a Palm Pilot. But in the near future I'll be able to throw away all these specialized gadgets. Instead I'll carry just the Handy 21, which will be able to download the appropriate configuration software and take on the functions of pretty much any device I want. The Handy 21 will contain a single Raw chip and several perceptual interfaces: cameras, small video displays and speech-based interfaces, including both speakers and microphones. It will use an antenna for communications and an analog-to-digital converter. The converter will be integrated on the same Raw chip, so that virtually all the functions for which we now buy special hardware will be accomplished by customizing applications directly into the chip's wires.

Our team has already built a compiler that can program applications directly into a simulator of the Raw chip. For example, we compiled a software radio application—which gives a personal computer the ability to function as an FM radio—to a 128-tile Raw chip [see "Communications Chameleons," on page 42]. Our results indicate that the application will run on the order of 10 times faster on the Raw chip than on any conventional microprocessor. Some of our students obtained a further 10-fold improvement by painstakingly translating the application without the aid of a compiler. But the real challenge is to develop a compiler that can approach the 100-fold improvement achieved by hand-customizing a chip's wiring to match the application.

If we are successful, the Raw chip could become a universal logic chip, a replacement for both general-purpose microprocessors and special-purpose microchips. We can say, with perhaps a bit of overstatement, that within a couple of decades there will be only three kinds of chips in the world: Raw chips, memory chips and, of course, potato chips. SA

ANANT AGARWAL co-directs the Raw project at the M.I.T. Laboratory for Computer Science. He is an associate director of the lab and professor of electrical engineering and computer science at M.I.T., where his research interests include computer architecture, compilation and software systems. He earned his Ph.D. in electrical engineering from Stanford University in 1987. The other members of the Raw project team are Michael Zhang, Michael Taylor, Mark Stephenson, Andras Moritz, Jason Miller, Albert Ma, Walter Lee, Sam Larsen, Jason Kim, Benjamin Greenwald, Matthew Frank, Rajeev Barua, Jonathan Babb and Saman Amarasinghe.

Further Reading

The M.I.T. Laboratory for Computer Science's home page (www.lcs.mit.edu) summarizes the work of the many research groups involved in the Oxygen project. The home page of the Spoken Language Systems group (www.sls.lcs.mit.edu/sls/) offers details on conversational interfaces and the speech-based applications that the group has developed. Information on software communications devices and wireless networking can be found at the SpectrumWare project's home page (www.sds.lcs.mit.edu/SpectrumWare/home.html). The Raw project's site (www.cag.lcs.mit.edu/raw/) includes a description of the Raw microprocessor as well as a list of publications by the project's team members. Contact information for the researchers is also available at the laboratory's Web site (www.lcs.mit.edu/contact/). For an overview of the future of information technology, a good source is *What Will Be,* by Michael L. Dertouzos (HarperCollins, 1997).