DICK POUNTAIN

THE TRANSPUTER STRIKES BACK

In 1985, Inmos launched the T414 transputer, the world's first microprocessor built for parallel processing. The concept was simple: Put a CPU, some memory, and four fast DMA/serial-communications links onto a chip, and then hook many chips together and make them work in parallel. Ganging up conventional microprocessors in this fashion inevitably leads to a communications bottleneck as the chips contend for control of the bus. With transputers, you can be sure that computation power and communications bandwidth remain roughly in balance as the network grows, since each extra chip adds a bit of both.

Implementing the concept required what was then state-of-the-art silicon technology. In its heyday, the 20-million-instruction-per-second T414 was the fastest 32-bit microprocessor on the market, and its 2-million-floating-point-operation-per-second cousin, the T800 (launched in 1987), held a similar honor for a while. Recently, however, new RISC and CISC processors from Sun, Mips Computer Systems, and Intel have obliterated this performance edge; for example, Intel's 860 easily outruns the T800, and some parallel computer vendors have switched to the 860 despite its conventional bus-based communications (see "Personal Supercomputing with the Intel i860," January BYTE).

Now Inmos strikes back with the IMS T9000, the first of a new generation of transputers. Once again pushing fabrication technology to its limits, Inmos has come up with a design that delivers 200 MIPS and 25 MFLOPS (at 50 MHz) and features greatly improved communications technology. Even better, while the T9000 uses CPU caching and superscalar parallel execution of multiple instructions (as does the 860), it doesn't require supersmart compilers or assembly language wizardry. Code compiled for older transputers should run on the T9000 at near-optimal speed.

A new, superscalar packet-switching version of the Inmos transputer—the T9000—should give RISC chips a run for their money

Equally revolutionary is the T9000's new packet-switched "virtual" communications system, which takes the responsibility for routing messages from the programmer's code and moves it to fast hardware, where it belongs. This innovation promises communication delays of only a few microseconds, even across networks containing 1000 or more processors. By contrast, routing done in T800 software exhibits delays of hundreds of milliseconds.

The T9000 is built on a 180-square-millimeter die and incorporates 2 million transistors. For comparison,

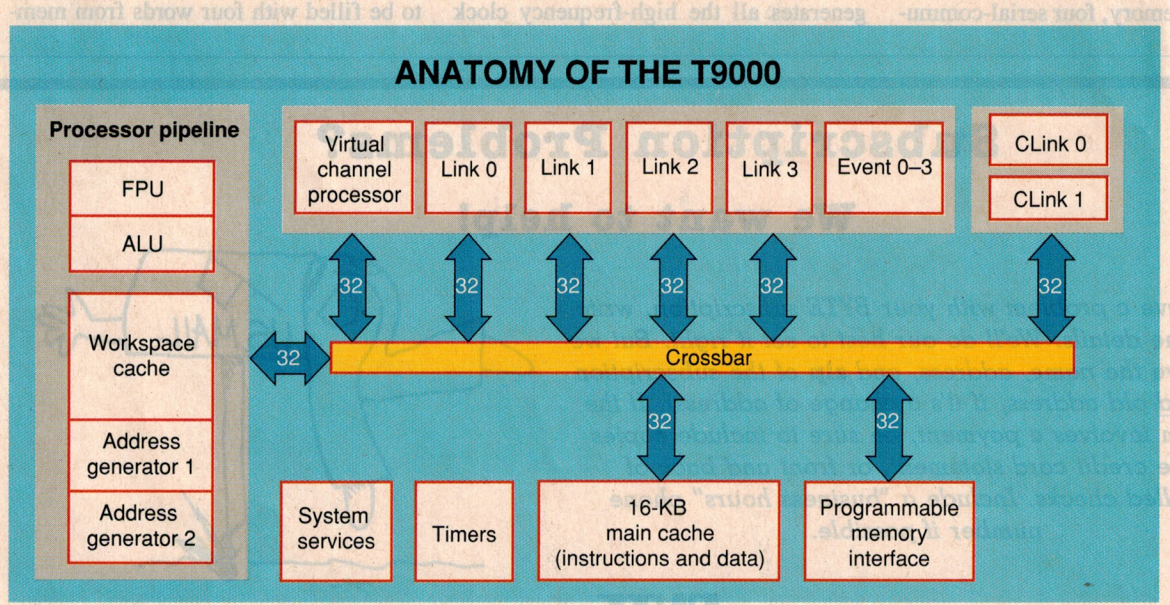

Figure 1: *With four independent data paths, Inmos's T9000 can compute and communicate simultaneously at high speeds.*

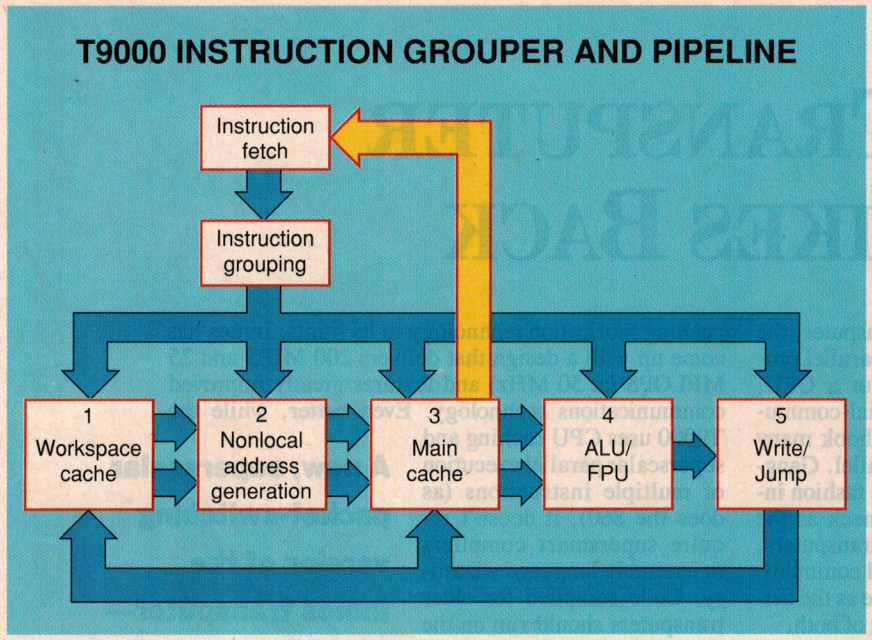

Figure 2: *The grouper automatically fills the five-stage pipeline.*

Intel's 486 and 860 each have about 1 million transistors. Inmos is using a new submicron CMOS fabrication process that employs three metal layers and tungsten plugs for interconnection. NEC recently announced a similar process (see Microbytes, May BYTE).

The T9000's Anatomy

The main functional units on the chip (see figure 1) are the 32-bit-integer processor core, a 64-bit FPU, 16 kilobytes of CPU cache memory, four serial-communication-link engines, a virtual channel processor (VCP), and a programmable memory interface (PMI). There are also two on-chip timers, four pairs of event channels for synchronizing internal processes with external events, and two control links that allow control signals to be sent between T9000s independently of the data links. The control links facilitate error handling, network configuration and analysis, bootstrapping, and resetting. An on-chip phase-locked loop generates all the high-frequency clock signals needed by the chip's subunits, so the chip needs only a single 5-MHz external clock signal.

Running down the center of the chip like a spine and connecting all these subunits together is a crossbar switch that controls four completely separate 32-bit data paths. These paths connect four subunits—the CPU, the VCP, the PMI, and the process scheduler—to the four banks of the main cache. Without multiple data paths, the T9000 couldn't achieve the bandwidths that it needs. The 200-MIPS CPU needs an instruction and data bandwidth of some 600 megabits per second to keep it busy, while the VCP needs to be fed at a rate of 120 Mbps when communicating with other chips on all four links. The 16-KB cache is organized into four banks of 4 KB, each with its own 32-bit address and data buses. The crossbar arbitrates and switches the buses among the subunits in such a way that all four cache banks can be accessed simultaneously in every cycle, providing a total cache bandwidth of 800 Mbps. Both the CPU and the VCP have multiple ports into the cache memory; the CPU has three read ports and one write port. This architecture lets the T9000's CPU compute at a peak rate without reducing the communications bandwidth (and vice versa).

The main cache holds data and instructions. Each of its banks maps one quarter of the T9000's 4-gigabyte address space using 256 four-word lines, each with a 26-bit fully associative tag. One of these lines is always kept empty, to be filled with four words from mem-

ory whenever a cache miss occurs. The cache then chooses another line at random to become the new empty line; if this line is dirty (i.e., its contents have changed), it is immediately written back to memory, a stratagem that Inmos calls "early write-back."

You can program the T9000 cache to act as 16 KB of on-chip memory, for embedded applications with no external memory system, or as a hybrid with 8 KB of cache and 8 KB of on-chip memory.

CPU Pipeline and the Instruction Grouper

Like previous transputers, the T9000 uses a stack-based CPU architecture. Three 32-bit registers called Areg, Breg, and Creg act as a push-down hardware stack for expression evaluation. Most instructions implicitly look for their operands on the stack; for example, add takes the top two stack items and leaves their sum on top of the stack. The FPU also uses a three-stage stack with 64-bit registers called FAreg, FBreg, and FCreg, but it is the CPU that transfers floating-point values (whose addresses are formed on the CPU stack) between memory and the FPU stack.

There are three other registers: Next Instruction, Operand, and Workspace Pointer. The Next Instruction register is just the program counter, but on the T9000, it works through a 32-instruction fetch-ahead buffer and a pipeline.

The Operand register is where instructions and operands are constructed. To reduce code size, the transputer family employs a frequency-encoded instruction format. Single-byte op codes select the most common operations, while multibyte op codes formed with special prefix codes select unusual ones.

The Workspace Pointer points to the block of local variables in on-chip memory, which form the workspace for the current process. Transputers use an on-chip process scheduler for multitasking, a function that operating-system software usually performs. Because there is so little state to save for each process, the context-switching time for the transputer scheduler is less than a microsecond.

The T9000 is even more efficient than previous transputers, thanks to a separate 32-word buffer, called the workspace cache, that holds the 32 most-recently accessed local variables. The workspace cache is triple-ported, allowing two reads and one write per cycle, and it writes through to the main cache. Most of the time, local values can be accessed entirely within the CPU, effectively in zero cycles, without reference to

INSTRUCTION GROUPING FOR a[i+1] := b[j+15] + c[k+7];

In a single cycle, the T9000 executes an entire group of instructions.

Operation	Meaning	Pipeline stage
Group 1		
ldl j	Load local *j*	1
ldl b	Load local address of *b*	1
wsub	Calculate address of *b*[*j*]	2
ldnl 15	Load value of *b*[*j*+15]	2, 3
Group 2		
ldl k	Load local *k*	1
ldl c	Load local address of *c*	1
wsub	Calculate address of *c*[*k*]	2
ldnl 7	Load value of *c*[*k*+7]	2, 3
add	Add top two stack values	4
Group 3		
ldl i	Load local *i*	1
ldl a	Load local address of *a*	1
wsub	Calculate address of *a*[*i*]	2
stnl 1	Store into *a*[*i*+1]	2, 5

external memory or even to the main cache. Because the workspace cache is a circular buffer that doesn't need to be flushed during context switches or external interrupts, it imposes no interrupts.

The workspace cache is actually the first stage of a CPU pipeline that lets the T9000 issue several instructions per cycle. This so-called superscalar architecture appears in several other advanced microprocessors, most notably the Intel 860. By judiciously choosing the right instructions, the T9000 can execute several instructions in parallel.

Figure 2 illustrates the T9000's five-stage pipeline structure. Within a single clock cycle, the first stage (the workspace cache) can fetch two local variables, thanks to its triple porting; the second stage can compute two addresses for nonlocal variables or array elements; the third stage (the main cache) can load two nonlocal variables; the fourth stage can execute an ALU or FPU operation; and the fifth stage can perform a conditional jump or a write to memory. Many arithmetic instructions have been sped up enormously compared to the T800; for example, an integer multiply now takes two to five cycles (compared to 38 for the T800), and a 64-bit floating-point multiply takes three cycles (compared to 27 for the T800).

A problem with previous superscalar designs was that the ordering of instructions, crucial to efficient pipelining, was left to the programmer or the compiler

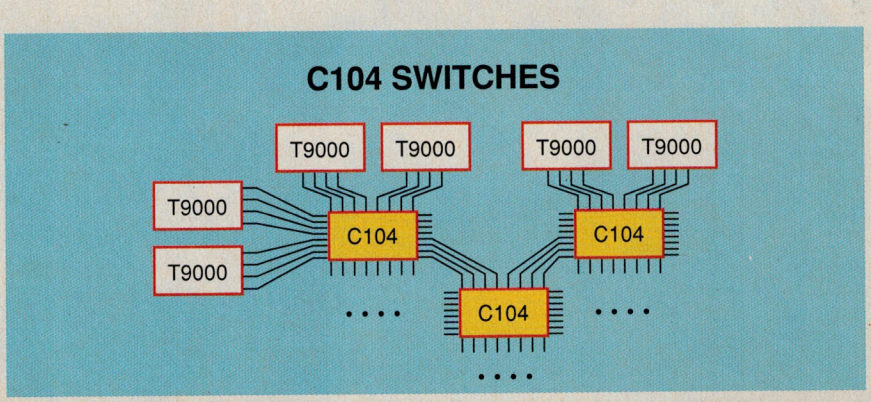

Figure 3: *The C104, with 32 links, can route 16 simultaneous conversations. It uses worm-hole routing and has a switching latency of just 700 nanoseconds.*

writer. Either way, tricky hand- or machine-generated code was needed to optimize instruction order for the pipeline. Smart compilers have been slow in arriving for the Intel 860, and that's been inconvenient for would-be developers.

Inmos's engineers chose a different strategy. The T9000's pipeline contains a hardware instruction grouper that automatically sorts the instruction stream into groups that can most efficiently execute together; the CPU actually executes one group per cycle. Consider this complicated assignment between array elements:

```
a[i+1] := b[j+15] + c[k+7];
```

The T9000 can execute this assignment in just three cycles by grouping the instructions, as shown in the table. The grouper can even cram in a couple of instructions from other surrounding routines at the same time; for example, the first group in the table does not use stages 4 and 5, so earlier instructions could occupy these stages.

Thanks to the grouper, compilers for the T9000 do not need to be supersmart, and code compiled for older transputers will run efficiently (typically 10 times faster than on a 20-MHz T800). This entire pipelining and caching system is transparent to the programmer, who just sees something that works like a T800, but faster.

Memory Interface and Protection

The PMI unit on the T9000 generates all the timing signals needed by DRAM-based memory systems, including multiplexed row/column addresses, refresh, and page-mode accesses. You can connect up to 8 megabytes of DRAM directly to the chip without any extra logic, and you can connect larger amounts with just some address and data buffers. The PMI automatically uses page-mode access for reads and writes from the same row address when page-mode DRAM is connected to it, so the T9000 can transfer a

whole cache line without resetting the row address. If 64-bit-wide DRAM is connected, the PMI will read an entire 16-byte cache line in two memory operations rather than four.

The PMI divides the total memory space into four banks (not necessarily the same as those allocated by the main cache), providing separate timing and decoding for each bank, and it sets the bus width to 8, 16, 32, or 64 bits. You can easily design mixed memory systems containing, say, DRAM, video RAM, ROM, and memory-mapped peripheral devices.

The transputer family was conceived primarily to support the model of concurrent processes communicating over channels, but there are other models of parallel computation, one of the most popular of which is shared memory. Shared-memory systems have an advantage when huge amounts of data need to be moved between processors. The T9000's PMI has a set of signals that control access to the memory system by external devices. System programmers can use these to implement shared-memory systems (or to control DMA-based coprocessors, such as graphics engines). Caching can present a problem for shared-memory systems, since one processor may write to a location that invalidates another processor's cache contents. The T9000 handles this situation by means of instructions that flush the caches when a synchronized shared-memory transfer takes place.

Also new with the T9000 is hardware support for shared resources, whether they be peripherals, such as printers, or processes running on a network server. A server automatically queues requests sent over several channels by clients. The server issues a `grant` instruction to connect the head of the queue to the resource when it becomes free. On older transputers, you had to handle this sort of shared access much less efficiently by using the `alt` instruction to poll all waiting channels.

The T9000 supports a degree of memory protection, although it falls short of full demand-paged virtual memory (that is slated for the next-generation chip in the mid-1990s). The protection scheme lets the T9000 check and translate addresses for four regions of memory that are sized independently. An operating system could treat these regions as stack, heap, code, and data spaces. You can implement the scheme within programs by launching special *P-processes* (via the `go-protected` instruction), each under the control of a normal parent process called a *supervisor*. A P-process can read from all regions but can only write to or execute code from regions for which it has permission to do so. Illegal access attempts by a P-process can trigger a hardware trap that returns control to the supervisor process (as does an error condition or the execution of a privileged instruction). After certain write traps, the supervisor can extend the size of a region and then restart the P-process. That's one way to implement dynamically allocated stacks.

Virtual Channels and Routing

The T9000, like its predecessors, has just four serial-communications links, although, at 100 Mbps each (in both directions), they are five times faster than in the T800. Older transputers communicated by a simple point-to-point byte stream. Programs were limited to just

eight channels per chip, and if you needed any more, you had to write explicit (and very tedious) code to multiplex your messages down these channels. The T9000's VCP changes all that. It's a packet-switching controller that chops messages up into 32-byte packets, with headers containing addressing information. The VCP interleaves packets from different processes down the same physical link, just the way Tymnet does, and these packets get reassembled at their destinations. Effectively, there's a continuous channel. Your programs can use as many of these "virtual channels" as they need, limited only by the maximum bandwidth of the links and your performance criteria.

Packet switching lets the VCP make far more efficient use of the links than the old transputers could. While any particular virtual channel is idle, messages from other processes can keep the same link busy. If one process sends a long message, shorter messages from other processes can be sent concurrently on the same link without waiting for the first to finish. The receipt of every packet is acknowledged by a return packet consisting of an empty header, and the sending process remains blocked until its last packet has been acknowledged. However, other processes can use the link while the blocked process waits, so the blocked process cannot monopolize the link.

For applications that use mainly nearest-neighbor communication (e.g., image processing), you can use T9000s by just connecting their links directly together. For programs that need more distant communication via a switched network, there is now a more efficient alternative: the T9000's companion routing chip, the IMS C104.

The C104 is a fast 32 by 32 nonblocking crossbar switch that can rapidly connect any one of its 32 links to any other. The topological possibilities are almost limitless. You could connect all four links of eight T9000s to a single C104, or to one link of 32 T9000s, or you can gang up C104s (see figure 3) to form even more complex topologies. You can also use the C100, a protocol conversion chip, to connect older transputers to the C104.

Implemented in the same advanced VLSI technology as the T9000, the C104 can have up to 16 messages passing between pairs of its links simultaneously, so for reasonably simple networks it has a negligible effect on the communications bandwidth. Its switching latency is typically 700 nanoseconds per chip traversed.

Unlike some routers, the C104 does not buffer whole messages or even packets, which would require a large amount of memory. Instead, it uses so-called worm-hole routing, where only the header of a packet is buffered on the chip. A routing decision is made according to the address information in the header, the crossbar is set, and then the body of the packet flows continuously through the chip from input to output. The header may well arrive at its destination before the tail has left the sender. As the tail of a packet passes through the network, the path closes up behind it, like a worm crawling through sandy soil.

The routing algorithm you use with the C104 is based on an interval-labeling scheme. You label each T9000 in the network with a unique address number, and every C104 stores the range (i.e., interval) of destination labels available. The C104 can then switch a packet to the cor-

rect link by making a single comparison, something like 6 ≤ header < 13. The label values are stored into C104 registers at boot time, using the T9000's control links.

Interval labeling is faster and requires far less on-chip memory than storing a full routing table, but it only works if the network is labeled in certain ways. Fortunately, there is always at least one complete labeling for any network, and a known algorithm that will find it. For many popular network topologies, including trees, grids, and hypercubes, there is a labeling that's guaranteed not to cause deadlock.

The C104 has several modes of operation that let you tune the performance of the network. In general, with single message headers, you won't know what actual route a particular message will take; however, you can tell the C104 to apply multiple headers, which get stripped off at each switch, to explicitly specify a route. Hot spots can arise if an application algorithm routes too many messages through the same switch; another C104 mode can eliminate hot spots by routing each message in two stages, via a randomly chosen "halfway house."

Building on a Solid Foundation

The T9000 displays the same elegance and rigor in its design as the original transputer and delivers far more power and flexibility. Like its predecessors, it could serve as a stand-alone workstation CPU, as a component for parallel supercomputers, or as an embedded real-time controller. Yet it begs for new uses; for example, the T9000 has enough communications bandwidth to capture digital TV or ISDN data streams in real time and still perform several computations on each datum. For more processing, you can add more T9000s in a scalable way. In applications such as real-time three-dimensional rendering for virtual reality, or as a color laser printer or fax engine, the T9000 should have a dramatic impact on the cost/performance ratio.

Unlike its predecessor, though, the T9000 is not entering the market cold. There is now a worldwide base of transputer expertise; software development tools such as ANSI C, C++, and Occam; and plenty of program code that will run straightaway on the new chip. With engineering samples of the T9000 due by the fourth quarter of this year, 1992 just might be the year of the transputer. ■

Dick Pountain is a BYTE consulting editor based in London. You can contact him on BIX as "dickp."