FEATURE ARTICLE

David Prutchi

Parallel Processing with Transputers

Uniprocessor technology is at a limit. Greater speeds require new technology. Instead of paying the increased costs, David advocates parallel processing. Find out how to build your own transputer.

When John Von Neuman established the basics for sequential computer architectures in 1947, he simplified his solution to its very basics—a series of primitive numeric manipulations executed on data stored in a memory system. Each manipulation was carried out, one at a time, by a centralized processor. Since then, major improvements in computational throughput have been achieved by using more ample instruction sets with wider data and address buses and by increasing system clock speed.

However, little can be done to speed up a system already running at full tilt. Current clock speeds bordering on 200 MHz already present difficult design and layout problems that heavily affect a system's flexibility, performance, and price. Further increases in clock speed may require sophisticated semiconductor materials (e.g., gallium-arsenide) and specialized interconnection technologies with prohibitive costs.

Ultimately, even if all other issues are solved, uniprocessor machines are limited by signals that can't travel faster than the speed of light across the finite dimensions of the processor.

Using multiple processors, tightly or loosely coupled, to share the workload achieves higher computational power without raw increases in speed. This strategy, called *parallelism*, can be exploited in three ways:

- Algorithmic—the algorithm is broken down into a pipeline of multiple processes

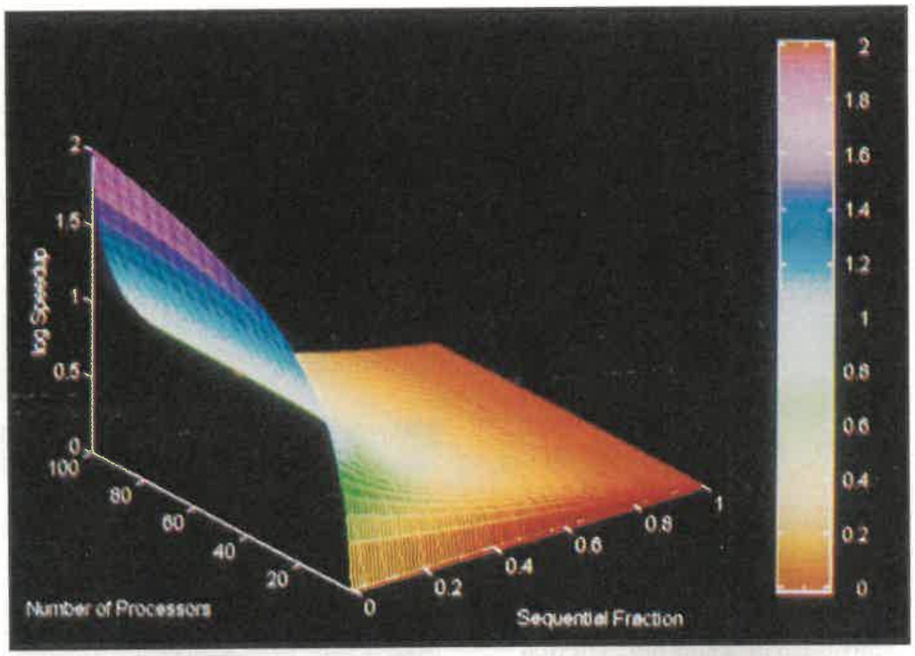


Photo 1—*A plot of the logarithmic (10) maximum speedup that can be achieved through an ideal parallel computer demonstrates that effective use of a multiprocessor machine can only be achieved through a drastic reduction in the percentage of time spent executing sequential code. For large sequential fraction values, even an infinite number of processors only achieves modest performance improvements.*

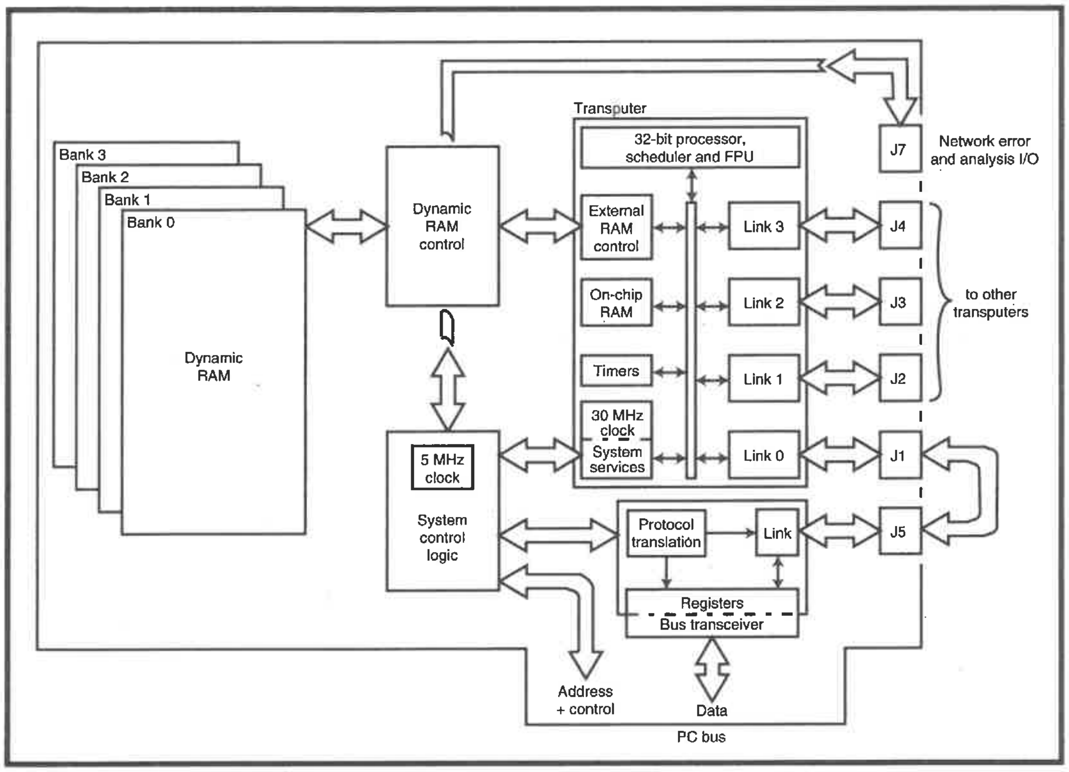


Figure 1—*The PC add-in transputer card features a 32-bit transputer, up to 4 MB of DRAM, and three high-speed links for connecting to an external transputer network. The card also offers a hierarchical network analysis structure to aid the development of multitransputer systems.*

- Geometric—the problem is broken down into a number of similar processes, each with a different subset of the total data to be processed. The processes communicate as they need access to data assigned to another process.
- Farming—the workload is "farmed out" by a master controller to other computing servers. The master dispenses new work to the servers as they become free.

Farming automatically balances the workload among the network servers regardless of network topology. It is limited only by the rate work can be dispensed and results handled. Unfortunately, farming is effective only when the problem can be divided into small, similar pieces, which represents only a small portion of the work demanded from a general-purpose computer.

Geometric parallelism introduces a major system design difference between Von Neuman sequential problem solving and parallel computation, namely the topology of the parallel processors' network. In principle, the processors need to be connected in such a way that the network topology somehow models the structures inherent to the problem. If a good model is found, the partitioning of the algorithm is simple to understand and implement.

Parallel machines based on specialized ICs have been developed experimentally for more than a decade. A number of parallel-processing computers and supercomputers are available, but only for large budgets. Lately, however, parallel processing chipsets have been designed that may launch the personal computer into affordable desktop supercomputing.

Intel and others offer parallel processing boards with a small number of scalable-architecture processors, such as the Pentium, i860, or i960. By sharing system resources, these processors achieve performances in the hundreds of MIPS range. At the same time, companies such as Chips and Technologies are introducing multiprocessor chipsets that address many of the hardware design issues for shared-memory multiprocessor systems. The CS8239 chipset, for instance, interconnects up to six '486 microprocessors to a fast, wide, nonmultiplexed bus that permits multiple masters.

Most of these efforts are inherently limited because bus-based, shared-memory computers (regardless of the number of processors they use) are limited in practical scalability by memory contention and bus bandwidth. Implementing distributed memory systems instead of shared memory schemes avoids most scalability limitations. However, processors with distributed memories require communication to effectively exploit geometric parallelism. This shift in the

Iptr—This instruction pointer acts as a conventional program counter. It points to the next instruction to be executed.

Wptr—This workspace or stack pointer points to a storage area of local variables.

Areg, Breg, Creg—These general-purpose registers form a push-down stack (Areg on top), so the transputer can use zero- and one-operand instructions. Because the stack is not large enough to store variables of lengthy operations, the transputer's assembly-language programming usually demands extensive use of load and store instructions, much like single-accumulator microprocessors.

Oreg—The operand register assembles the operands used by direct instructions.

Error Flag—This flag approximates a traditional overflow flag. Once triggered, however, it remains set until explicitly cleared. The state of this flag is presented to the transputer's Error pin, and the location of an errant transputer may be located in a multitransputer network.

Halt-on-error Flag—When set, this flag causes the setting of the error flag to be interpreted as a fatal error. The whole system comes to a complete stop.

Table 1—*The transputer has six internal registers and two flags which define the state of a sequential process.*

architectural paradigm can block development of parallel processing desktop supercomputers.

Already, there's a battle to capture the desktop supercomputer market between Inmos (now owned by SGS-Thomson Microelectronics) and microprocessor industry giants. Since 1985, Inmos has offered a *transputer*, which is a microprocessor that gets its power from the radically different philosophy that underlies its design, although it barely performs à la pair with Intel and Motorola processors.

Transputers communicate with other transputers in parallel processing networks using minimal interconnection and communications overhead. The same level of parallelism can be executed on a network of devices as within a single transputer executing virtual concurrent processes through hardware-scheduled sharing of the processor time.

After looking at the principles of parallel processing hardware and software, this article presents first-generation transputers in detail and a simple PC add-on implementation. A brief peek at Occam, the transputer's native language, is followed by a look at second-generation transputer products and the possibilities for desktop parallel computers that can stand up to the performance of multimillion dollar supercomputers.

PARALLEL PROCESSING

Different approaches to the design of parallel-processing computers have been identified to break the processing-speed limitations of sequential architectures. Essentially, these approaches aim to overcome the Von Neuman bottleneck of Single-Instruction Single-Data (SISD) computers.

Parallel architectures attempt to gain power by performing the same mathematical operation on a number of data elements simultaneously (Single-Instruction Multiple-Data—SIMD) or by assigning multiple unique tasks to many processors in parallel (Multiple-Instruction Multiple-Data—MIMD). SIMD is typical for vector or array processors while MIMD is the basis for more flexible parallel computers because it can work efficiently over a wide range of granularity.

There is even a hybrid of these two architectures called Single-Program Multiple-Data (SPMD). A copy of the same program runs on each processor, even though the programs are not synchronized at the instruction level.

The efficiency of each approach may be estimated using Amdhal's law, a mathematical formula which assumes that a computing process can be divided into a sequential and parallel portion [1]. Besides the parallel operations which may be distributed over a number of processors, there remains a sequential portion, comprising at least the sequential communications between the processes. The sequential component limits the efficiency of a parallel machine.

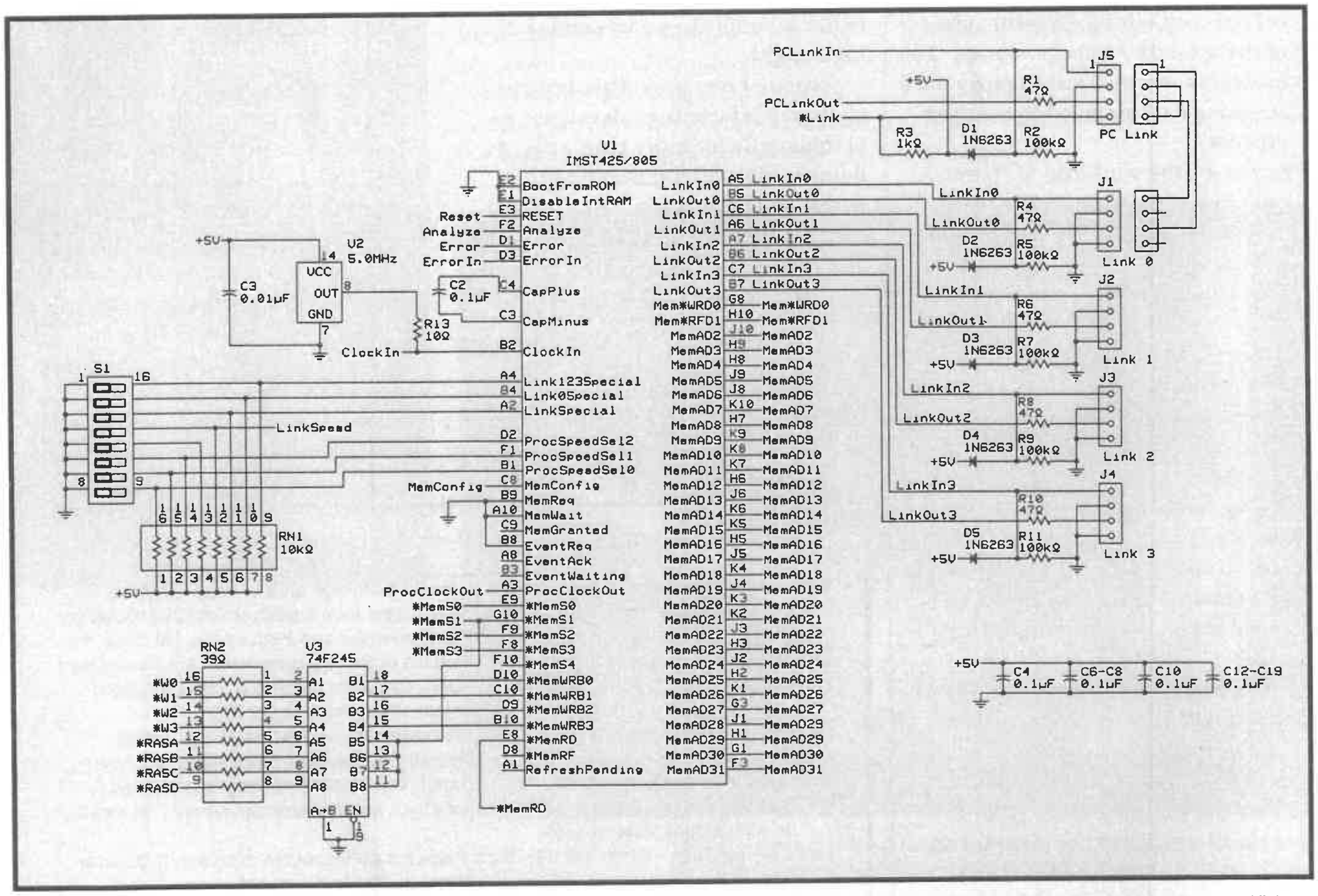


Figure 2—*The heart of the add-in card is a T425 or T805. These transputers communicate with the host PC and other transputers via four high-speed bidirectional serial links. The processor clock is internally generated by a PLL oscillator locked to IU2's 5-MHz clock. Processor and link speeds are selected through DIP switch bank S1.*

Under ideal conditions where no overhead for communications and synchronization is required, the maximum speedup attainable by converting a sequential program to a parallel implementation is expressed by Amhdal's law as:

$$\text{Speedup} = \frac{T_1}{T_n} \qquad (1)$$

where T_1 is the run time of the sequential version, while T_n is that of the parallel implementation, and n is the number of processors. T_n is determined by the fraction of the total time spent executing sequential code s and by the number of available processors as defined by:

$$T_n = sT_1 + \frac{T_1(1-s)}{n} \qquad (2)$$

Photo 1 displays speedup under ideal conditions as a function of the number of processors and percentage of time spent executing sequential code. The point is obvious—the major enemy to speedup is sequential processing. For large values of s, time loss is so significant that an infinite number of processors only achieves modest performance increases.

In more realistic conditions where workload is not perfectly balanced and communications and synchronization requires overhead, there's an additional toll to the theoretical speedup.

In recent years, the interpretation of Amhdal's law has been frequently challenged. More conservative views contend that Amhdal's law applies to all processing systems and thus describes a more general limitation on the performance of any programmed system above a certain level of complexity. Under this view, a good sequential or parallel system design should loosen the bounds imposed by Amdhal's law to the point where the system becomes feasible [2].

A more radical view disputes the validity of Amdhal's classical basis by arguing that program execution time rather than problem size is constant. From this perspective, speedup is achieved by having the software design the hardware it needs. This idea recently led to the design of a massively reconfigurable logic computer in which the hardware—676,000 gates in 52 FPGAs—is tailored through the software to exactly fit the algorithm. Virtual Computer Corporation, the developers of this machine, expect speedups well beyond those predicted by Amdhal's law.

In any case, Amhdal's law conveys the "catch" of parallel computing—different kinds of multiprocessing systems suit different kinds of applications. The designer must decide what is best for the problem at hand [3].

As a first step, assess optimal granularity. Although some algorithms run more efficiently at fine-grain level (simultaneously executing many different microinstructions such as move, add, compare, write, etc.) where multiple parallel processors can be accommodated with ease, the organizational overhead of communications and synchronization may consume the speed gains of parallelism.

However, using coarse-grain parallelism (simultaneously executing subroutines) doesn't ensure success. Tasks that could be executed in parallel may remain locked within subroutines, so some processors remain idle for large amounts of the computing time.

To best develop a multiprocessor system, the designer needs to thoroughly understand the application and then develop a well-structured program that is highly modular and easy to partition. Many engineers experienced in parallel programming first develop and debug the algorithm in a single-task version before attempting to deal with communications, synchronization, and resource sharing.

The structured program should be carefully analyzed to identify how to best distribute the tasks. To achieve the desired speedup, optimization aims to balance the load between parallel processes, while reducing communications and synchronization requirements. If performed correctly, this step provides a clear view of the hardware and software topology and the level of granularity that best exploits parallelism for your application.

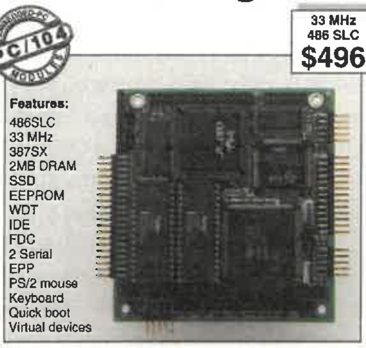
The designer should also identify how data and code distribution can be

YOU DON'T GET TO THE TOP BY THINKING SMALL, SO WHY MESS AROUND WITH INFERIOR TOOLS. ADOPT TEAM PARADIGM AND TAP INTO THE FINEST TOOLS, BACKED BY UNLIMITED FREE TECHNICAL SUPPORT.
TO BE CONTINUED...

#107

Sets the Pace
in PC/104
CPU and DAS
Technologies

33 MHz
486 SLC
$496



Features:
486SLC
33 MHz
387SX
2MB DRAM
SSD
EEPROM
WDT
IDE
FDC
2 Serial
EPP
PS/2 mouse
Keyboard
Quick boot
Virtual devices

CMi486SLC-1 Fully Integrated PC-AT with Virtual Device Support

When placing your order, mention this ad and receive **a 387SX math coprocessor FREE!**

200 kHz
12-bit DAS
$498

Features:
200 kHz
12-bit A/D
16SE/8DI
Scan
Burst
Multiburst
1K CGT
Triggers
FIFO buffer
Data marker
12-bit D/A
16-bit DIO
+5V operation

DM5408-2 200 kHz Analog I/O Module with Channel-Gain Table

Make your selection from:

9 cpuModules™

XT, SuperXT™, 386SX, 486SLC, 486SXLC2, 486SX, 486DX, and 486DX2 processors. SSD, 8MB DRAM, RS-232/422/485 serial ports, parallel port, IDE & floppy controllers, Quick Boot, watchdog timer, power management, and digital control. Virtual devices include keyboard, video, floppy, and hard disk.

7 utilityModules™

SVGA CRT & LCD, Ethernet, keypad scanning, PCMCIA, intelligent GPS, IDE hard disk, and floppy.

18 dataModules®

12, 14 & 16-bit data acquisition modules with high speed sampling, channel-gain table (CGT), sample buffer, versatile triggers, scan, random burst & multiburst, DMA, 4-20 mA current loop, bit programmable digital I/O, advanced digital interrupt modes, incremental encoder interfaces, opto-22 compatibility, and power-down.

Real Time Devices USA
200 Innovation Boulevard • P.O. Box 906
State College, PA 16804-0906 USA
Tel: 1 (814) 234-8087 • Fax: 1 (814) 234-5218
FaxBack®: 1 (814) 235-1260 • BBS: 1 (814) 234-9427

RTD Europa
Budapest, Hungary
Fax: (36) 1 212-0260

RTD Scandinavia
Helsinki, Finland
Fax: (358) 0 346-4539

RTD is a founder of the PC/104 Consortium and the world's leading supplier of PC/104 CPU and DAS modules.

#108

DIP Switch			Processor Speed (MHz)	
S1-8 (ProcSpeed Select0)	S1-7 (ProcSpeed Select1)	S1-6 (ProcSpeed Select2)	T400	T425/T805
0	0	0	20.0	20.0
0	0	1	22.5	22.5
0	1	0	25	25.0
0	1	1	30	30.0
1	0	0	35	35
1	0	1	Invalid	Invalid
1	1	0	17.5	17.5
1	1	1	Invalid	Invalid

Table 2—*Processor-speed selection is available in discrete steps. Clock frequency is programmed through DIP switches S1-6 through S1-8 up to the maximum rated frequency for a particular device.*

carried out independently. This often results in a network which can't be described as a pure processor farm, geometric array, or algorithmic pipeline. A simulation of a certain physical phenomenon may have a geometric processor array to model the physical system, each of which feeds a pipeline simulating a local physical process. These in turn may own a farm of processors which care for math operations requiring minimal interaction.

TRANSPUTER BASICS

The transputer is a RISC computer on a chip, complete with a high-speed CPU, memory, external memory controller, and four full-duplex communication links. Some models also include an on-chip floating-point unit (FPU). The transputer's architecture achieves optimal virtual and true concurrent processing under Occam.

Occam enables transputers to be described as a collection of processes operating concurrently and communicating through channels. It implements the Communicating Sequential Processes (CSP) model of parallel computing. It considers a parallel program to be the same as a finite number of sequential processes which execute concurrently while exchanging messages over message channels.

As a processing unit, the transputer's integer CPU calculates address information and presents the FPU with data. The CPU efficiently supports the Occam model of concurrency and communication. A reduced number of instructions form the transputer's instruction set and make concurrent processing as efficient as possible.

A stack-based architecture uses on-chip RAM as a conventional microprocessor uses its registers. Concurrent tasks map their "registers" to specific workspace in the on-chip RAM. Table 1 describes the internal registers and two flags which define the state of a running process.

Switching between concurrent tasks is as simple as switching a pointer to the correct workspace. An operating-system kernel is built into the transputer to execute multitasking under direct hardware control. Hence, any number of concurrent processes can be executed together, sharing the processor time. Hardware-controlled scheduling eliminates the need for external software kernels and speeds context switching to 0.6 µs.

The hardware-process scheduler automatically sleeps processes waiting for channel I/O and wakes them at I/O completion. The instruction set also includes an Alternative command (ALT) which makes a process dormant until it receives an alternative enabling event. Two interval timers and time-out support also keep processes dormant until they're needed.

Communication between concurrent processes takes place through channels when both the input and the output processes are ready. This message-passing channel construct lets processes share data and become synchronized. Communication between processes on the same transputer takes place through local-memory channels. When processes run on different transputers, communication takes place through channels implemented on the high-speed serial links. Each link is a fast (20-Mbps), asynchronous, full-duplex channel.

In addition to the links, the built-in memory controller communicates with external memory and peripherals. The memory controller expands the address space off chip, and can directly control up to 4 GB of DRAM. It also maps I/O space for interfacing with other peripherals. Since these modules operate simultaneously, asynchronous communications between processes demands minimal overhead.

notMemS0 (*MemS0)	Address latch enable
notMemS1 (*MemS1)	Row address strobe (*RAS), which is selected during all memory and I/O access operations
notMemS2 (*MemS2)	Row/column address multiplexing
notMemS3 (*MemS3)	Column address strobe (*CAS)
notMemS4 (*MemS4)	Not used
notMemRf (*MemRf)	Not used
notMemRd (*MemRd)	Not used for memory access; used to control the I/O PAL
notMemWr (*MemWr)	Write control to all memory chips

Table 3—*Strobe lines are used for external memory interface.*

TRANSPUTER SPECIES

Transputers operate with clock rates of 15, 20, 25 and 30 MHz, and different transputer families fulfill the requirements of different markets and applications.

The T2*xx* comprises a family of 16-bit transputers. These "baby transputers" have 64 KB of memory space and lack many features of their 32-bit counterparts. Because of their low cost (~$80 in singles for the 30-MHz model), they are especially attractive for parallel-processing embedded applications which do not require high-precision arithmetic or extensive memory.

The T2*xx* is often used as a signal-acquisition controller or preprocessor within systems that use more powerful transputers as their main processors. As you'll see later, integrating other types of transputers is possible because the internal register pointer architecture and link protocols work with different word lengths.

The T4*xx* transputer family contains 32-bit microprocessors, and implements all the features of the T2*xx* family. However, the T425 has four serial links and 4 KB of on-chip SRAM, while the T400 has only two serial links and 2 KB of on-chip SRAM.

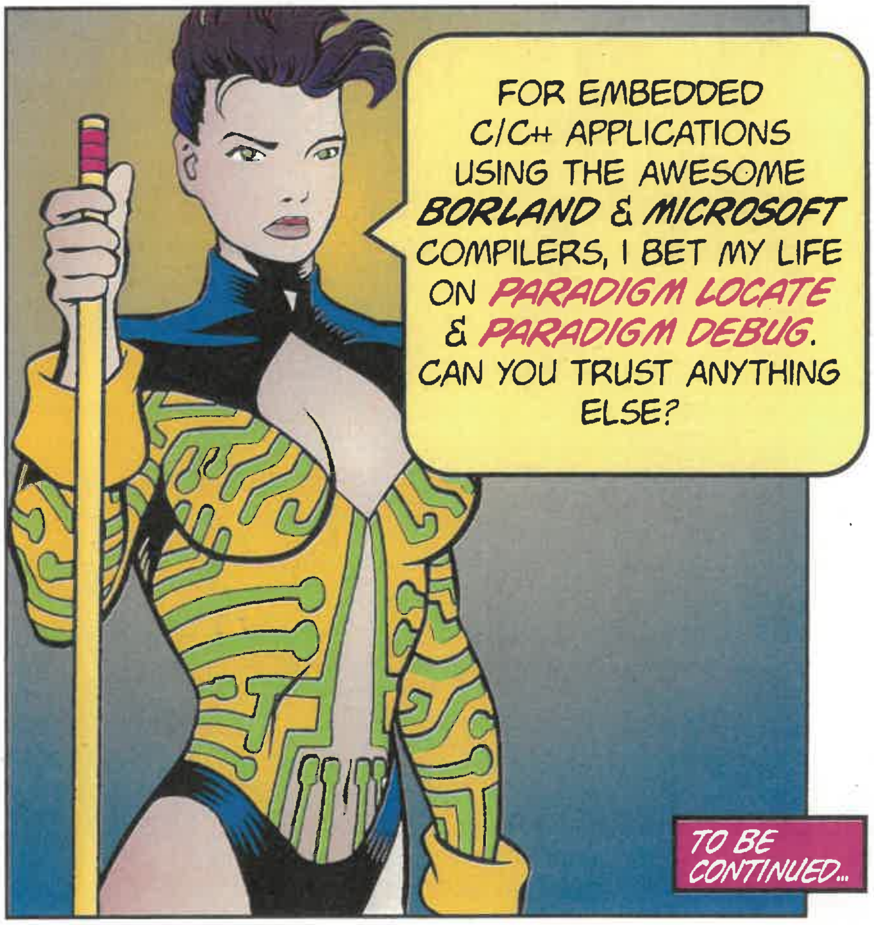
The T8*xx* is essentially the same as the T4*xx* family, except it has an on-chip floating-point coprocessor. The T800, recently replaced by the improved T805, is pin-compatible with the T425. But, with the aid of its coprocessor, it is capable of delivering over 4.3 MFLOPS and 30 MIPS peak (@ 30 MHz) on its own.

The T9000 second-generation transputer is the latest addition to the transputer family. This new device sports numerous hardware enhancements which increase speed and support advanced operating systems while maintaining upward compatibility with the T805 instruction set.

More importantly, each of the T9000 links is connected to multiplexing hardware. Communications among processes in separate transputers takes place along as many channels as required. Shared physical links are software transparent. Much more flexible parallel programs can be implemented to exploit the full power of the CSP model.

The design of the T9000 is a truly remarkable leap beyond the T8*xx* family. Its advanced features include a pipelined superscalar processor that delivers more than 150 MIPS and 20 MFLOPS, on-chip high-speed cache RAM, and 100-Mbps links. However, its shortage of silicon and support hardware and software have been a problem. Although the T9000 was announced in 1991, it only recently started coming out of fab. Single 20-MHz units retail at $450.

Inmos also offers a number of interesting support ICs that make it easy to design and interface flexible transputer networks. The IMS C012 is an IC which converts the serial transputer protocol into parallel format and vice versa. The IMS C004 link provides a crossbar switch between a maximum of 32 transputer links. It is cascadable and enables reconfiguration


FOR EMBEDDED C/C++ APPLICATIONS USING THE AWESOME BORLAND & MICROSOFT COMPILERS, I BET MY LIFE ON PARADIGM LOCATE & PARADIGM DEBUG. CAN YOU TRUST ANYTHING ELSE?
TO BE CONTINUED...

#109

of the transputer network's topology under software control.

ICs with similar functions are available for the T9000's 100-Mbps links, including the IMS C104, a high-performance routing chip that interconnects T9000 transputers to form a full-blown packet-switching network. T9000s can be used in combination with any of the first-generation transputers by using an IMS C100 link converter IC, which translates 20-Mbps links to the 100-Mbps links of their second-generation counterparts.

TRANSPUTER ADD-IN BOARD FOR THE IBM PC

In 1987, Inmos introduced a T414 transputer add-in board for the IBM PC as the B004 model [4]. It was developed as an Occam engine hosted by the PC. In addition, it could accelerate computationally intensive tasks for the PC either alone or in conjunction with a transputer network. The B004 still supports transputer didactic and development environments for the PC.

Although the B004 performs well for unsophisticated applications, the introduction of the T8*xx* family made many T4*xx*-based products obsolete. Figure 1 offers an improved circuit, inspired by the simplicity of the B004 circuit, to illustrate the design of practical transputer systems. This simple circuit remains compatible with the original B004, but also accepts the powerful members of the T8*xx* family.

Based on this figure and some help from Inmos's transputer data book [5], it is relatively easy to develop more advanced systems. Figures 2–7 should enable you to build a fully functional transputer PC add-in card. Although it's a bare-bones approach, this design features compatibility with Inmos and third-party software; up to 4 MB of local DRAM; compatibility with T400, T414, T425, T800, and T805 transputers; DIP-switch selection of processor and link communication speeds; and DIP-switch memory configuration.

Ideally, transputer boards should be constructed using four-layer PCB technology to ensure noise-free reliable operation. As with every board designed for high-speed operation, proper impedance matching and termination, extensive power-rail and ground-plane decoupling, as well as path-delay equalization are required [6].

THE T4*xx*/T8*xx* TRANSPUTER IC

As shown in Figures 2 and 4, multiple VCC pins minimize inductance within the IC, and all must be connected to a well-decoupled power rail. Similarly, all GND pins must be connected to the board's ground plane. C2, a 0.1-µF ceramic capacitor, must be soldered directly between CapPlus and CapMinus to appropriately decouple the internal clock supplies. A 0.1-µF ceramic decoupling capacitor between the VDD and ground planes of the PCB should be placed near the transputer's socket to aid in decoupling the power supply to this IC.

Clock design is simple since all first-generation transputers, regardless

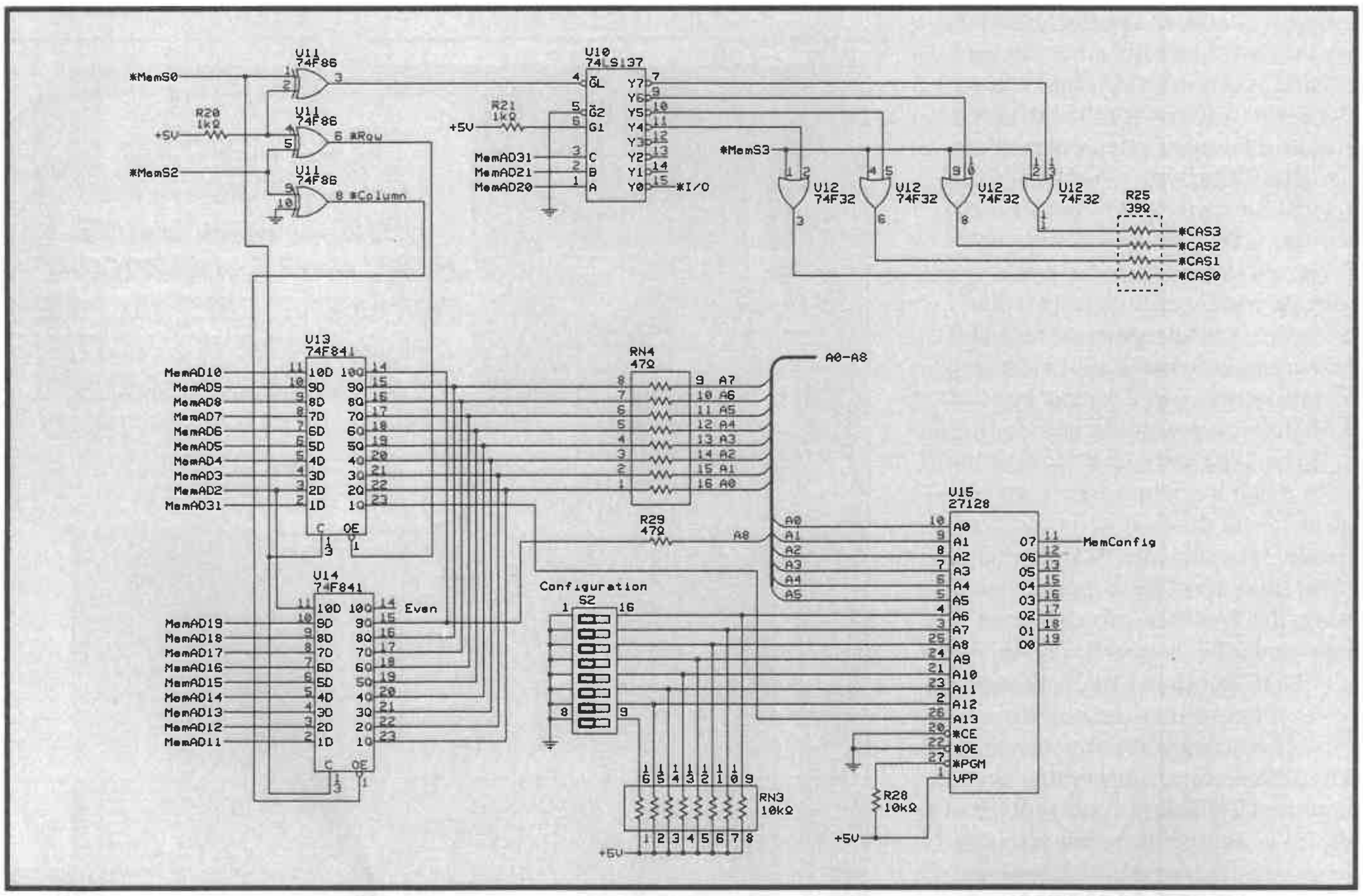


Figure 3—*External memory address and control-signal decoding is accomplished through this circuit. The actual memory configuration (strobe use, timing, etc.) is automatically loaded from EPROM IU15 into the transputer on processor reset.*

of device type, use a 5-MHz clock. The processor's high-frequency clocks are internally generated, which eases design and layout constraints so it's easier to construct a well-behaved board.

U2, a 5-MHz crystal oscillator, produces a stable 5-MHz clock signal. Processor speed, up to the maximum rated speed for a particular transputer, is selected through DIP switches S1-6, S1-7, and S1-8 as shown in Table 2.

MEMORY AND I/O SYSTEMS

The memory system implemented in the board is capable of supporting up to 4 MB of DRAM. This capacity is divided among four banks of 1 MB, each of which is implemented using four 256 KB × 9 DRAM SIP modules. For simplicity (unlike the original B004), parity-error checking has not been implemented.

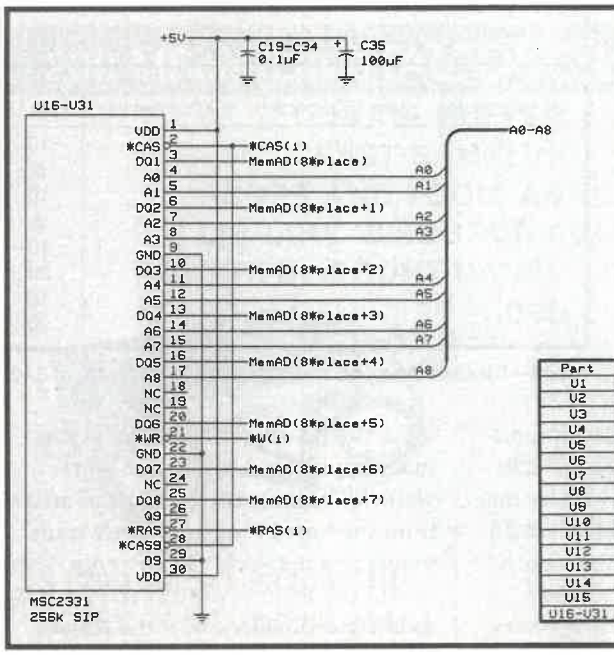
The transputer has a built-in programmable memory interface which greatly simplifies the design of memory. This interface's programming determines the configuration of the external memory cycle required for a specific type of memory. In the board, a configuration corresponding to a given selection is entered through a DIP-switch bank and is translated into the appropriate memory configuration by U15, a 27128 EPROM (see Figure 3).

A detailed explanation regarding external transputer system memory configuration can be found in the transputer's data sheet. From this, you can configure almost every variation of the memory system you may want to implement in your card.

In addition, `S706BETA`, the public-domain software tool used to design the EPROM code for this add-in card, helps you develop custom RAM configurations to be loaded in U15. `S706-BETA` and `PROMLOAD`, another freeware utility, help you develop bootable EPROM code for embedded applications.

In the add-in card, the EPROM is cycled by the decoded address bus A0–A5 and by the Scan/*Read line into two corresponding phases. During the Scan phase, all *MemS(i) strobes are held logic high. This state renders both 74LS841 latches transparent. When this happens and *MemS2 asserts the *Row line, the logic high state of MemAD31 selects the EPROM's upper 8 KB. Zeros programmed in this area ensure that MemConfig is held low so the transputer loads an external configuration.

During the Read, MemAD31 remains low, selecting the EPROM's lower 8 KB. Different memory cycle configurations are stored in 128 segments, each of which is 64 bytes long. The desired segment number is user-selected through the configuration



Power Table

Part	+5V	Gnd
U1	C2,C5,E10,H4	A9,C1,G2,G9,J7
U2	14	7
U3	20	10
U4	24	12
U5	20	10
U6	20	10
U7	20	10
U8	24	12
U9	14	7
U10	16	8
U11	14	7
U12	14	7
U13	24	12
U14	24	12
U15	28	14
U16-U31	1,30	9,22

Figure 4—*Up to 16 256-KB × 9-bit DRAM modules are available to the transputer as external memory. External memory is distributed as four banks of 1-MB × 8-bit DRAM.*

Be a hero – put *Team Paradigm* to work on your next embedded x86 design and reap the rewards you deserve. *Team Paradigm* has the ability to deliver all the embedded system pieces from Intel to AMD, C/C++ or assembly language, and all the Borland/Microsoft development tools. Plus **Paradigm DEBUG** for your favorite in-circuit emulator and real-time operating system.

So forget about making excuses and instead enlist *Team Paradigm* for your current or next x86 project. We deliver the finest embedded x86 development tools, backed by unlimited free technical support. No one is more serious about your success than Paradigm Systems. Call today and get the details before you waste any more of your precious time.

PARADIGM 'Nuff said.

Proven Solutions for Embedded C/C++ Developers

1-800-537-5043

Paradigm Systems
3301 Country Club Road, Suite 2214, Endwell, NY 13760
(607) 748-5966 / FAX: (607) 748-5968
Internet: 73047.3031@compuserve.com

TO BE CONTINUED.

#110

DIP-switch bank S2. The correct memory-configuration sequence results through the cycling of lines A0–A5. Table 3 shows how to use the transputer strobe lines.

The transputer system's RAM is mapped into negative space. Therefore, RAM selection occurs during the logic-high cycles of MemAD31. Selection of the appropriate bank is done through MemAD20 and MemAD21. When enabled by the address latch enable signal (*MemS0) and *CAS strobe (*MemS3), column-address selection signals (*CAS(i)) corresponding to each of the memory banks are decoded by U10, a 74LS137, which receives MemAD20,21,31 as inputs.

DIP Switch			Link Speed (Mbps)	
S1-3 (LinkSpecial)	S1-1 (Link123Special)	S1-2 (Link0Special)	Link 0	Links 1,2,3
0	0	0	10	10
0	0	1	5	10
0	1	0	10	5
0	1	1	5	5
1	0	0	10	10
1	0	1	20	10
1	1	0	10	20
1	1	1	20	20

Table 4—*Transputer link speeds are selected through switches S1-1, S1-2, and S1-3.*

During refresh cycles, MemAD31 is logic low, thus disabling all *CAS(i) lines. RAM refresh-only is implemented on the board to take care of DRAM-refresh operations without disturbing I/O.

Only the subsystem port is mapped into the I/O area. This area resides at a positive address, which implies that *CAS(i) generation is disabled by the logic low of MemAD31 during I/O operations. I/O space is limited to the first megabyte of positive space because both MemAD20 and MemAd21 have to be low for I/O operations under the present implementation. The Even/*Odd selection line is the latched version of MemAD2, which appears during *MemS2's falling edge. Figure 4 shows connection of the DRAM modules to the decoding logic and the transputer.

CONTROL SYSTEM

The transputer has a number of incoming and outgoing system-level control signals, which include processor reset (Reset), bootstrap control (BootFromROM), and facilities for error analysis (Error, ErrorIn, and Analyze).

The falling edge of Reset initializes the transputer, triggers the memory-configuration sequence, and starts the bootstrap routine. In this card, since the BootFromROM line is permanently grounded, the transputer waits for booting instructions to arrive from the host PC or from other transputers in a network. On powerup, U9e causes the logic in the 22V10 PAL (U8) to unconditionally assert the Reset line.

The error and analysis signals aid in developing and troubleshooting new designs. They become especially handy when debugging multitransputer networks. Asserting the Analyze line causes the processor to halt whenever it reaches specified breakpoint conditions, complete outstanding link transactions, and place special status values on the transputer's registers for debugging.

The Error pin conveys the state of the transputer's internal error flag ORed with the ErrorIn line. Internal errors are caused by conditions such as arithmetic overflow, divide by zero, array-bounds violation, or through direct software selection of the internal error flag. In a multitransputer network, the ErrorIn and Error pins of a number of transputers can be daisy-chained to halt the network, making the status of each processor available for probing by a master transputer. In the transputer where the error originates, the error flag is not cleared by a processor Reset, so its location can be identified. Executing the `testerr` instruction clears the flag and allows for normal system operation.

The add-in board's control signals are mapped to the PC's I/O space. By doing so, the PC can reset and analyze a network of transputers connected to the card's up or down subsystem ports. This feature prevents errors in the transputer network from flowing into the PC, so the PC can always be ready to reset and upload the add-in board.

Input control signals are Up-Reset (*UPR) and Up-Analyze (*UPA). These signals arrive from modules of higher hierarchy than that of the add-in transputer card while Subsystem Error (*SSE) signals arrive from modules of lower hierarchy. *UPR generates an unconditional Reset signal to the onboard transputer and link adapter. *UPA signals the Analyze input. *SSE can be read by the onboard transputer at Occam address #200000000 (absolute address zero) as a logic 1 in the LSB of the word.

Output control signals generated by the board are Subsystem Reset (*SSR) and Subsystem Analyze (*SSA).

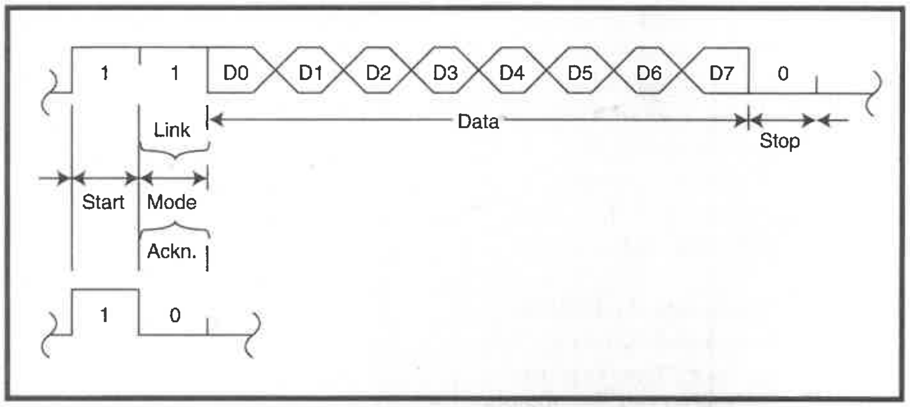


Figure 5—*In the protocol for data transmission over high-speed serial links, each byte transmitted through the LinkOut line of a channel is preceded by a start bit and followed by a 1 bit and a stop bit. After each transmission of a byte, the sending device waits for an acknowledge signal. This acknowledge signal, received through the LinkIn line of the transmitting device, is formed by a start bit followed by a 0 bit.*

These signals control transputers placed in lower hierarchies than those of the PC add-in board. Line *SSR can be asserted by the onboard transputer by writing a 1 to the LSB of absolute address zero. Similarly, *SSA can be asserted by writing a 1 to the LSB of absolute address 4 (Occam address # 20000001).

When the board is in the enhanced mode, the PC and onboard transputer can control an attached network of transputers. To do so, *SSR and *SSA are asserted through control signals PCUpReset and PCUpAnalyze generated by the PC. This approach prevents an error-generating network from interfering with the resetting and booting of the PC's add-in card. The status of the onboard transputer Error pin is placed in the up port as the Up-Error signal *UPE.

If you want, the hierarchy arbitration can be changed by deasserting the *System line so that an error received from an attached transputer network through Down-Error signal *DNE is forwarded to the PC as a *UPE. *UPR and *UPA signals received from the up port are buffered and forwarded to the down port as a Down Reset (*DNR) and Down Analyze (*DNA).

The system and subsystem control logic is implemented in U8, a 22V10 PAL. The Subsystem, PC, Up and Down Reset, Analyze, and Error signals are brought out of the add-in card through edge connector J7 for easy connection to an external transputer network.

For normal operation, the *System line must be jumpered to ground (J7-2 to J7-1), and the PC control lines must be connected to the up port: *PCR to *UPR (J7-5 to J7-11), *PCA to *UPA (J7-7 to J7-13), and *PCE to *UPE (J7-3 to J7-9).

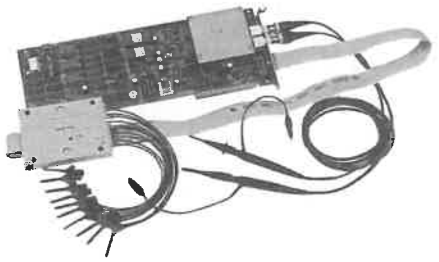
5-, 10-, AND 20-MBPS SERIAL LINKS

Communications through a transputer link is carried out through a simple protocol which supports the synchronized communication requirements of Occam. This protocol transmits an arbitrary sequence of bytes, which interconnects transputers with different word lengths.

PC-Based Instruments

200 MSa/s DIGITAL OSCILLOSCOPE

HUGE BUFFER
FAST SAMPLING
SCOPE AND LOGIC ANALYZER
C LIBRARY W/SOURCE AVAILABLE
POWERFUL FRONT PANEL SOFTWARE



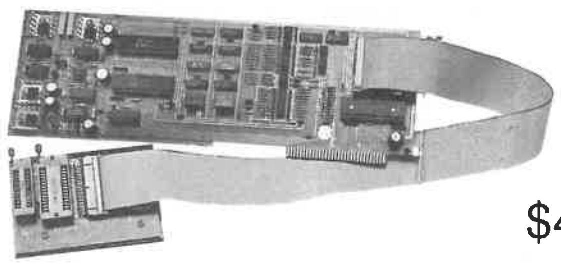
DSO Channels
2 Ch. up to 100 MSa/s
or
1 Ch. at 200 MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

Logic Analyzer Channels
8 Ch. up to 100 MHz
4K or 64K Samples/Ch
Cross Trigger with DSO

$1799 - DSO-28204 (4K)
$2285 - DSO-28264 (64K)

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
PIC
etc..



$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

· up to 128 Channels
· up to 400 MHz
· up to 16K Samples/Channel
· Variable Threshold Levels
· 8 External Clocks
· 16 Level Triggering
· Pattern Generator Option

$799 - LA12100 (100 MHz, 24 Ch)
$1299 - LA32200 (200 MHz, 32 Ch)
$1899 - LA32400 (400 MHz, 32 Ch)
$2750 - LA64400 (400 MHz, 64 Ch)

Call (201) 808-8990

Link Instruments
369 Passaic Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786

#111

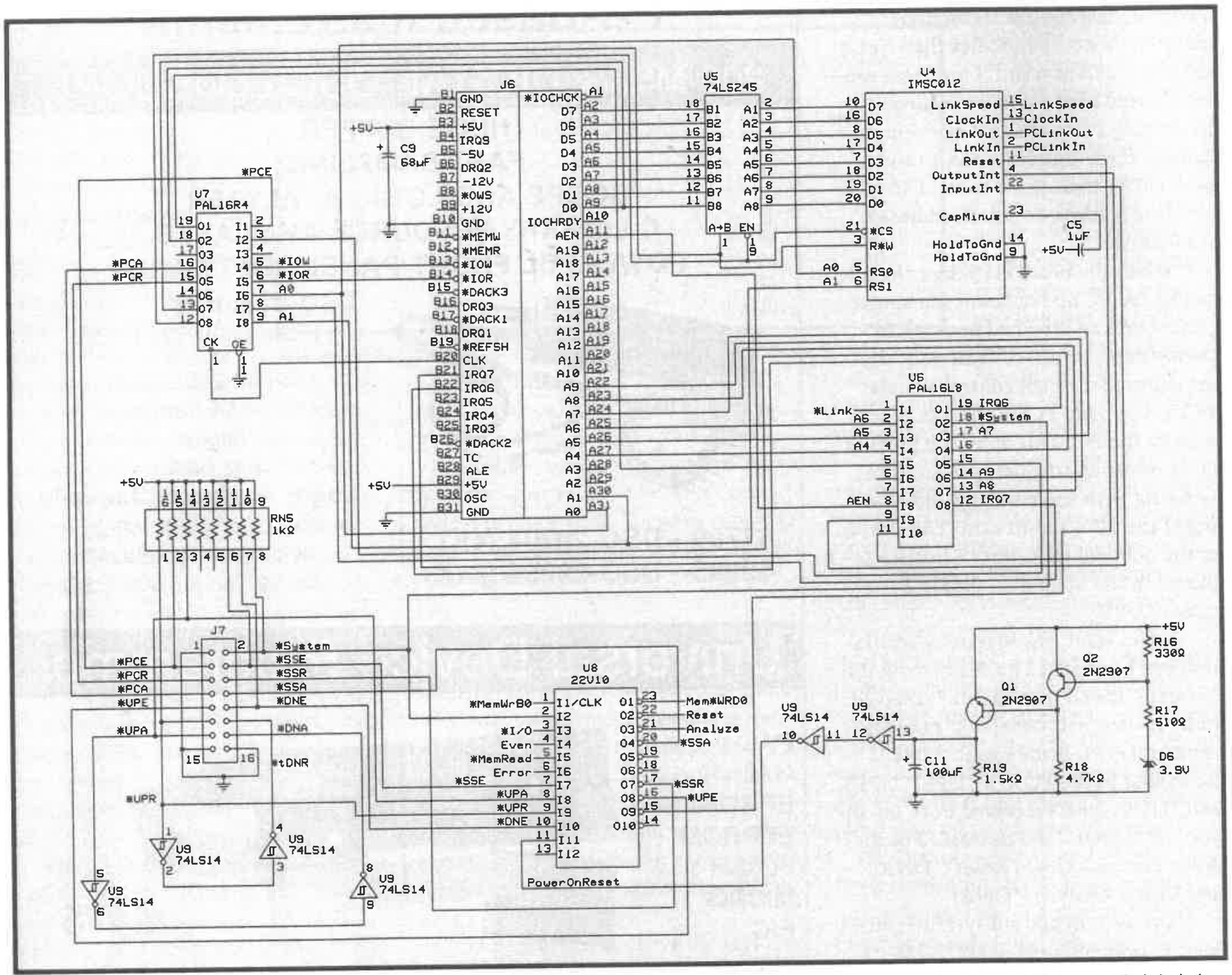


Figure 6—*The interface between the transputer's link 0 and the PC data bus is performed though U4 and bus transceiver U5. Glue logic, as well as error and analysis logic, is implemented in PALs U6, U7, and U8.*

As shown in Figure 5, the eight bits of each byte transmitted through a channel's LinkOut line are preceded by a start bit and followed by a 1 bit. The data bits are then followed by a stop bit. After each byte's transmission, the sending device waits for an acknowledge signal from the receiving device before proceeding. This acknowledge signal, received through LinkIn on the transmitting device, is formed by a start bit followed by a 0 bit.

Transmitted bytes and acknowledgments are multiplexed on the various signal lines. As long as a process is waiting for the sent data and there is room in the data buffer, acknowledgment is sent immediately on data latching. The transmission is carried out without delays between successive data bytes.

Transputer links are designed for electrically quiet environments. Keep the direct interconnection between transputers to about 1'. The link lines are fully TTL-compatible, so industry-standard line drivers and receivers can extend the distance of these links.

Interface design is simple because all first-generation transputers use a 5-MHz oscillator with PLL for reference, and no phase reference is required. Separate transputers interconnected through the same link may thus operate from independent clocks, each of which may be running at a different frequency.

Link speeds can be selected between 5-, 10-, and 20-Mbps by programming switches S1-14, S1-15, and S1-16. Table 4 shows how link zero can be set independently from links 1, 2, and 3, which must be programmed to the same speed.

PC INTERFACE

The host PC communicates with the transputer through one of its serial links. As far as Occam is concerned, this maps the PC as a process connected through a channel implemented on that dedicated link.

As shown in Figure 6, translation between Inmos's serial-link protocol and the PC parallel data bus is performed through U4, an IMS C012 link adapter IC. Parallel data transfers between the adapter and the PC occur through U5, a 74LS245 bus transceiver, under the control of the logic programmed into PALs U6 and U7.

U6 and U7 decode the address bus, data direction controls, and address/

data bus demultiplexing, translating them into an appropriate timing sequence for the link adapter. The PALs' logic also arbitrates the master, slave, and subsystem hierarchies for the process, so Occam recognizes the PC.

Connection of the PC interface to the transputer's link 0 is done through jumper cables between the PC link and link 0 edge connectors (J1 and J5). Asserting *Link low enables communications with the PC bus. The link speed for the IMS C012 must match that of link 0, which is set through switch S1-4. When this switch forces line LinkSpeed low, the link adapter IC operates at 10 Mbps. When high, it operates at 20 Mbps.

The logic in U6 makes software written for Inmos's B004 compatible with this board by placing it in the same location in the PC's I/O address space (150h–163h).

To experiment with true parallelism, additional processors can be added to the transputer system. Since additional transputers may be booted through the serial links, only one transputer in the network needs direct connection to the PC bus.

In additional cards, *Link must be high (achieved by removing the jumper between pins 2 and 4 of connector J5). Glue logic which interfaces the board to the PC may be omitted (or the ICs removed from their sockets). All links of the transputers are available for network interconnection by removing the connections between the PC link and link 0 connectors J5 and J1.

Extra cards use the PC only as a power source and do not need to share the same motherboard or power source. They may reside in an external chassis capable of supplying appropriate power and cooling or may occupy a slot in a PC.

TESTING THE BOARD

Testing your newly assembled transputer hardware is simple through PC-Check, a freeware utility package which determines the types, versions, functionality, and topology of all transputers in a network. As shown in Figure 7, PC-Check also identifies the speed of the link connected to the PC and performs a basic test of each

```
a)
A:\>check | mtest

Using 150 check 3.0
  # Part rate Mb Bt [Link0 Link1  Link2   Link3] RAM,cycle
  0 T425b-20 0.17 0   HOST   ...    ...     ...   4K,1+1022,4

b)
A:\>check | ftest

Using 150 check 3.0
  # Part rate Mb Bt [Link0 Link1  Link2   Link3] (Ftest)
  0 T425b-20 0.17 0   HOST   ...    ...     ...   (OK)
```

Figure 7—*(a) An example of a PC-Check test of the transputer PC add-in card shows that* `CHECK.EXE` *determines the types, versions, and topology of all transputers in a network, and reports on all available RAM. (b)* `FTEST.EXE` *uses the piped output from* `CHECK` *to establish the functionality of each transputer in the*

transputer's internal and external memory.

In Figure 7a, CHECK.EXE identified a T425b as connected to the host through link 0, running at 20 MHz, and the sole transputer in the network. The pipe into MEMTEST.EXE reports 4 KB of 1-cycle memory (internal memory) and 1022 KB of 4-cycle RAM. Piping the output of CHECK.EXE into FTEST.EXE (Figure 7b) executes a full functional test of each network transputer. Each device which passes the battery of tests receives an "OK" message.

When CHECK.EXE runs, it leaves a vestigial routing trail at each node which sets a default communications path from the host to any transputer in the network. Through this path, LOAD.EXE, another program in PC-Check, can load programs on any transputer node. The package also includes a hex monitor and a simple method for configuring the topology of a network connected through one or more IMS C004 software-controlled crossbar switch ICs.

To run your own programs, use IServer as an interface between the PC host and the transputer network. IServer, a public domain package, allows the PC to reset, boot, and analyze a transputer network. It also gives the transputer network access to various PC resources such as file I/O, keyboard input, and CRT output.

OCCAM PROGRAMMING

Transputers can be programmed in most popular programming languages, but its special properties are best exploited by Occam since the transputer directly supports the Occam model of concurrency and communication.

Occam syntax uses indentation to indicate program structure. Specialized folding editors are used to write a program. A folding editor represents a large text block in a single named fold line marked by three dots. Folds can be nested to any level and created before their contents are written, allowing the structure of a process to arise as the design's skeleton.

Occam describes a system as a collection of distributed concurrent processes that communicate with each other through channels. Occam consists of three primitive processes which are combined to create larger processes:

- Output—`c!e` outputs expression `e` into channel `c`
- Input—`c?v` inputs channel `c` into variable `v`
- Assignment—`v:=e` assigns expression `e` to variable `v`.

These primitives are joined by three types of constructors: sequential (SEQ), parallel (PAR), and alternate (ALT). Statements or subprocesses contained within the context of SEQ execute in sequence, while those under PAR execute concurrently. In contrast, ALT has the processor execute whatever component process is ready first. Programs can be built in the conventional way by employing variables, assignments, mathematical and logical expressions, and with standard constructs such as IF, WHILE, and FOR.

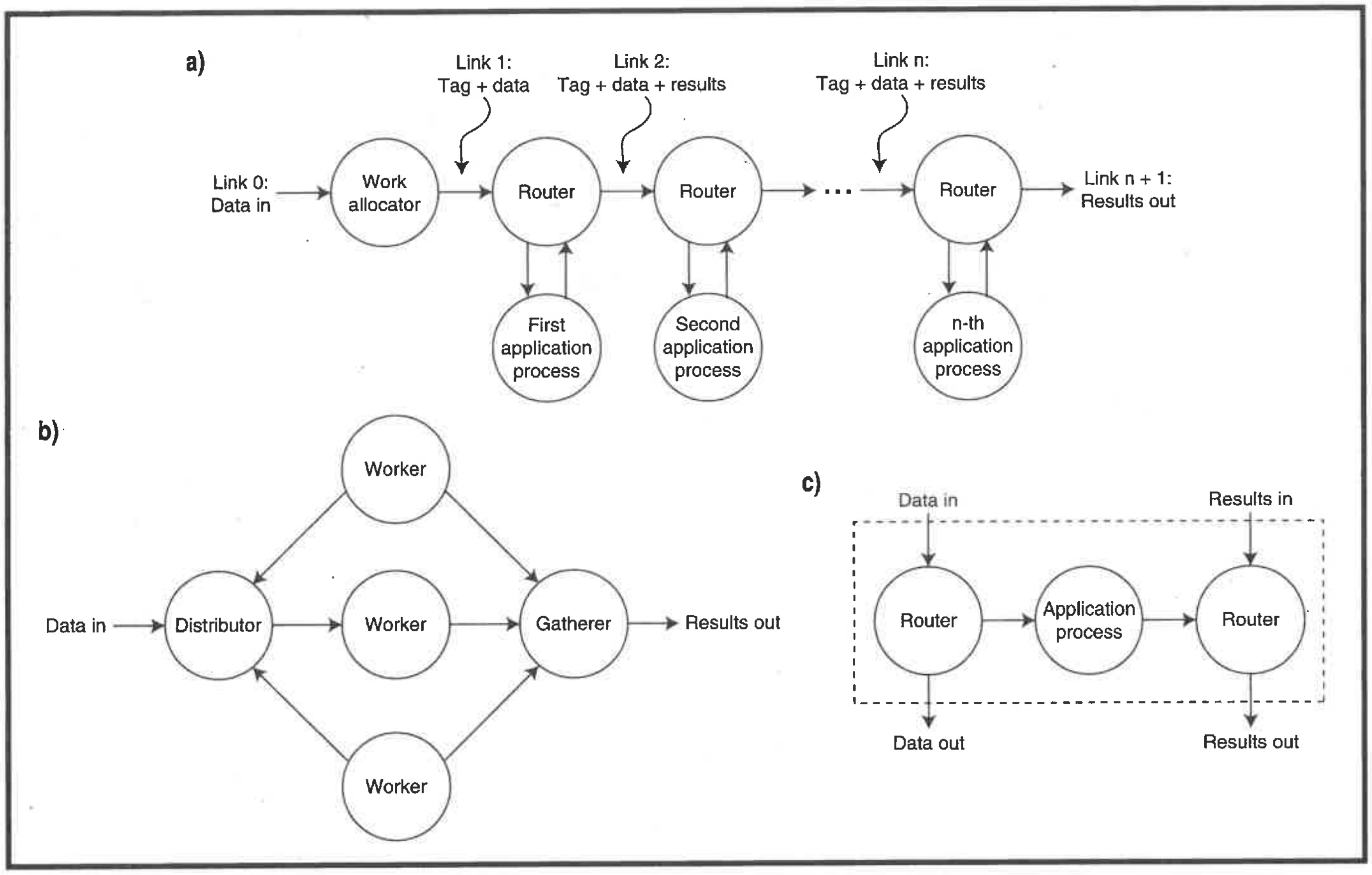


Figure 8—*The most simple topology supported by the PAR replicator syntax is a pipeline (a). A spaceline system is slightly more complex, using a distributor process to hand out data to concurrent worker processes, and a gatherer process to collect and combine results (b). Input and output of these structures can be at one end by introducing an additional router to achieve bidirectional communication (c).*

Each Occam channel provides a communication element between two concurrent processes. Information is sent via a channel in one direction only without message buffering. A channel is perceived as a read-only element to a receiving process and a write-only element to a transmitting process. Communication is synchronized and takes place when both the inputting and outputting processes are ready.

The ALT construct elegantly selects between executing parallel processes. It waits for input from a group of input channels and then executes the process attached to whichever one becomes active first. If more than one input arrives simultaneously, ALT chooses to act based on assigned process priority and the implementation of the program. Its choice ensures that no channel becomes permanently locked out.

PAR's power lies in its simple way of replicating arrays of similar processes. As shown in Figure 8, the most simple topology supported by the PAR replicator syntax is a pipeline. Each stage performs data processing and passes data and/or results on behalf of other processes.

A spaceline system is slightly more complex. A distributor process hands out data to concurrent worker processes, and a gatherer collects and combines results. In addition, since channels are available in both directions, a router process can be introduced so input and output can be at one end of a pipeline.

The interconnection between processes is limited only by the number of available links. While this doesn't pose a problem for processes executing concurrently on a single transputer, it does when the transputer's physical links are used as Occam channels. Then, simultaneous connectivity between transputers is limited to the four high-speed serial links.

This arrangement still allows for many useful topologies. Binary trees and H-trees provide almost unlimited extensibility. Two-dimensional grid arrays are also extensible and present improved communication and reduced path lengths between elements. The next level of architecture is the n-dimensional cube or binary n-cube. With $n = 2$ and 3, the topology is that of a square and a cube, respectively. A 4-cube or basic hypercube has 2^4 transputers and achieves an almost ideal configuration for connectivity and channel-path length.

A hypercube can mimic other topologies. Processing can occur as a number of independent trees, 2-cubes, or 3-cubes just by preventing communications between selected nodes. Since it is topologically identical to a torus (an improved grid array), the hypercube is flexible, making it a favorite architecture for multitransputer networks.

By using multiple transputers within a single node to extend the number of links available to each node, higher-order hypercubes are possible. A two-transputer supernode has six available links and can be used as a 6-cube. A four-transputer supernode forms an 8-cube, and so on.

SECOND-GENERATION TRANSPUTERS

Despite a wide variety of topologies, implementation of universally parallel algorithms isn't possible using first-generation transputer hardware and software because they comply only partially with the CSP parallelism model. The discrepancy exists because message exchange between parallel processes in different transputers is limited by the transputer's four communications links.

The T9000 second-generation transputer provides a remedy. Besides hardware enhancements that increase operation speed and support advanced operating systems, the T9000 provides greater connectivity. Each link is connected to multiplexing hardware.

Communications among processes in separate transputers takes place along as many channels as required. Shared physical links are transparent to software, so true CSP parallel programs can be used.

This capability is achieved through a separate communications processor. It transmits messages that multiplex a large number of virtual links along each physical link. Virtual links support one Occam channel in each direction. Messages are transmitted as a sequence of 32-byte data packets. A header which precedes the packet routes the packet through the network and identifies the destination virtual link on the remote transputer.

The new packet communications hardware simplifies programming because software systems then don't need process allocation. Different allocations can be used for different networks, and they can change dynamically to optimize performance.

Moreover, the compiler can make the allocation, removing configuration details from the program. This is a vital step toward development of a general-purpose parallel computer.

To form a full-blown packet-switching network, one or more IMS C104 high-performance routing chips interconnect T9000 transputers. Each 'C104 provides 32 bidirectional 100-Mbps links. The header of each packet arriving on a link input determines the link a packet should output to as soon as the physical link is free.

The basis for optimal packet routing is an algorithm called *interval labeling*. An interval is a continuous set of header values and is associated with an output channel. The header of an incoming packet is limited to within a single range, and the packet is directed toward the appropriate interval. Most common network topologies have stable and efficient labeling schemes.

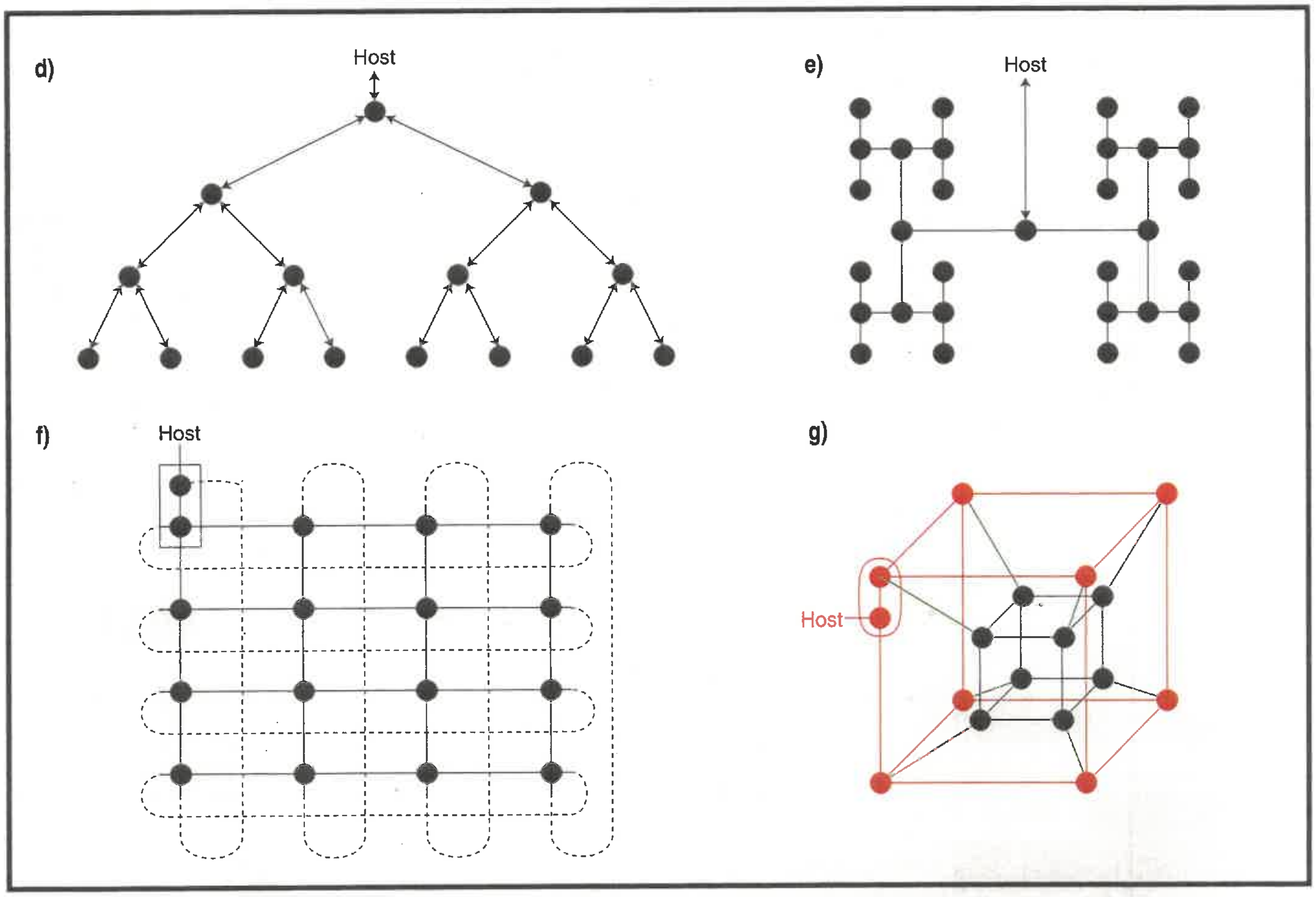


Figure 8 (continued)—*Communications between processes running in separate transputers (shown as dots) are limited by each transputer's four high-speed links. Useful topologies such as a binary tree (d), an H-tree (e), and a processor mesh (f) can be implemented with ease. Connecting the free links of the mesh (broken lines) results in the very powerful topology known as the basic hypercube (g).*

In the T9000, parallelism starts to be exploited within the processor by using a pipelined superscalar architecture. The processor core executes up to eight instructions on each 20-ns clock cycle. Inmos even has dedicated on-chip hardware which controls the flow of multiple instructions through the pipeline, making it possible to run existing T8xx code at blazing speeds.

Another improvement is the move toward a cached architecture supported by 16 KB of on-chip cache RAM. This much RAM is often enough to support lightning-fast embedded applications without external RAM. The T9000 still supports external memory through a programmable memory interface, but it also has a 64-bit data bus with high data-transfer rates for cache-line refill. In addition, the new interface supports four independent banks of memory, each of which can be individually programmed.

The T9000 also provides trap handling and protected processes. The former enables error processing through software before control is returned to the process in which the error occurred. The latter supports secure programming and debugging in embedded systems, which protects processes and the operating system from other errant programs executing concurrently.

Second-generation transputers can be interfaced to previous-generation transputers by using the IMS C100 link converter IC mentioned earlier. In an advanced real-time embedded application, for example, T805 transputers can acquire and preprocess sensor-array signals, while a network of T9000s simultaneously executes computationally intensive processing, analysis, sensor fusion, and control portions of the program. Output post-processing and actuator control could use the less powerful T2xx or T4xx transputers as embedded controllers.

YOU—PART OF THE PC REVOLUTION?

Throughput of PCs and workstations has increased dramatically in recent years, and uniprocessors are quickly approaching the limits of circuit designs using current fabrication technologies. It is safe to assume that keeping up with increasing demands for computational speed will result in increased parallel processing.

To a certain extent, it is already happening. Take a look under the hood of your PC. Dedicated processors are on almost every add-on card, providing intelligent graphics, I/O, and cache controllers. New multimedia cards integrate DSPs, signal and image compression and decompression, and stand-alone media controllers. Some companies take this trend one step further by offering dual-Pentium motherboards, and multi-i860 add-in boards for the PC.

The shift toward truly parallel machines will be gradual—the industry-standard PC won't be massively parallel for a number of years. Yet, within the next two years, there will be more and more ads for small-scale superscalar and parallel PCs in which a small number of Pentium (or P6, etc.) processors run concurrently within a bus-based, shared-memory system.

The migration to distributed-memory systems may happen only when the scalability limitations of shared-memory systems are reached. Whose chips will dominate then? With the aid of today's low-cost parallel processors, you may well be one of the pioneers to exploit personal computing's next revolution. ■

David Prutchi has a Ph.D. in Biomedical Engineering from Tel-Aviv University. He is an engineering specialist at Intermedics, and his main R&D interest is biomedical signal processing in implantable devices. He may be reached at davidp@mails.imed.com.

REFERENCES

[1] G. Amdhal, "Limits of Expectation," *International Journal of Supercomputer Applications*, **2(1)**, 88–94, 1988.

[2] B. Christianson, "Amdahl's Law and the End of System Design," *Performance Evaluation Review*, **19(2)**, 30–32, 1991.

[3] L. Kleinrock and J.H. Huang, "On Parallel Processing Systems: Amdhal's Law Generalized and Some Results on Optimal Design," *IEEE Transactions on Software Engineering*, **18(5)**, 434–447, 1992.

[4] S. Ghee, *IMS B004 IBM PC Add-in Board*, Inmos Technical Note 11 (72TCH01100), 1987.

[5] Inmos, *The Transputer Databook*, 3rd. Ed., 1992.

[6] D. Prutchi, "Designing Printed Circuits for High-Speed Logic," *INK*, **42**, 38–43, 1994.

SOURCES

INMOS
1000 East Bell Rd.
Phoenix, AZ 85022
(602) 867-6100

Transtech Parallel Systems
20 Thornwood Drive
Ithaca, NY 14850-1263
(607) 257-6502
Fax: (607) 257-3980

Computer System Architects (CSA)
100 Library Plaza
15 North 100 East
Provo, Utah 84606-3100
(801) 374-2300
Fax: (801) 374-2306

Transtech and CSA offer transputer hardware and distribute many software packages. These include n-parallel Prolog interpreter by Paralogic; Parallel C, C++, Fortran, Pascal for the Transputer by 3L; a C transputer toolset from Logical Systems; and the Professional OCCAM Toolset, ANSI C Toolset, ANSI Fortran 77 Toolset, and Inquest Debugger from INMOS.

Parallel-processing software is also freely available through the Internet. Contact ftp://unix.hensa.ac.uk and ftp://ftp.inmos.co.uk.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#64 NOVEMBER 1995

ANALOG DESIGN

Rediscovering Analog Computers

A Transputer-based Parallel Computer

Developing a Virtual Hardware Device

Power-Line Modems

$3.95 U.S.
$4.95 Canada