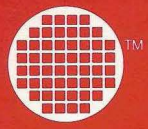
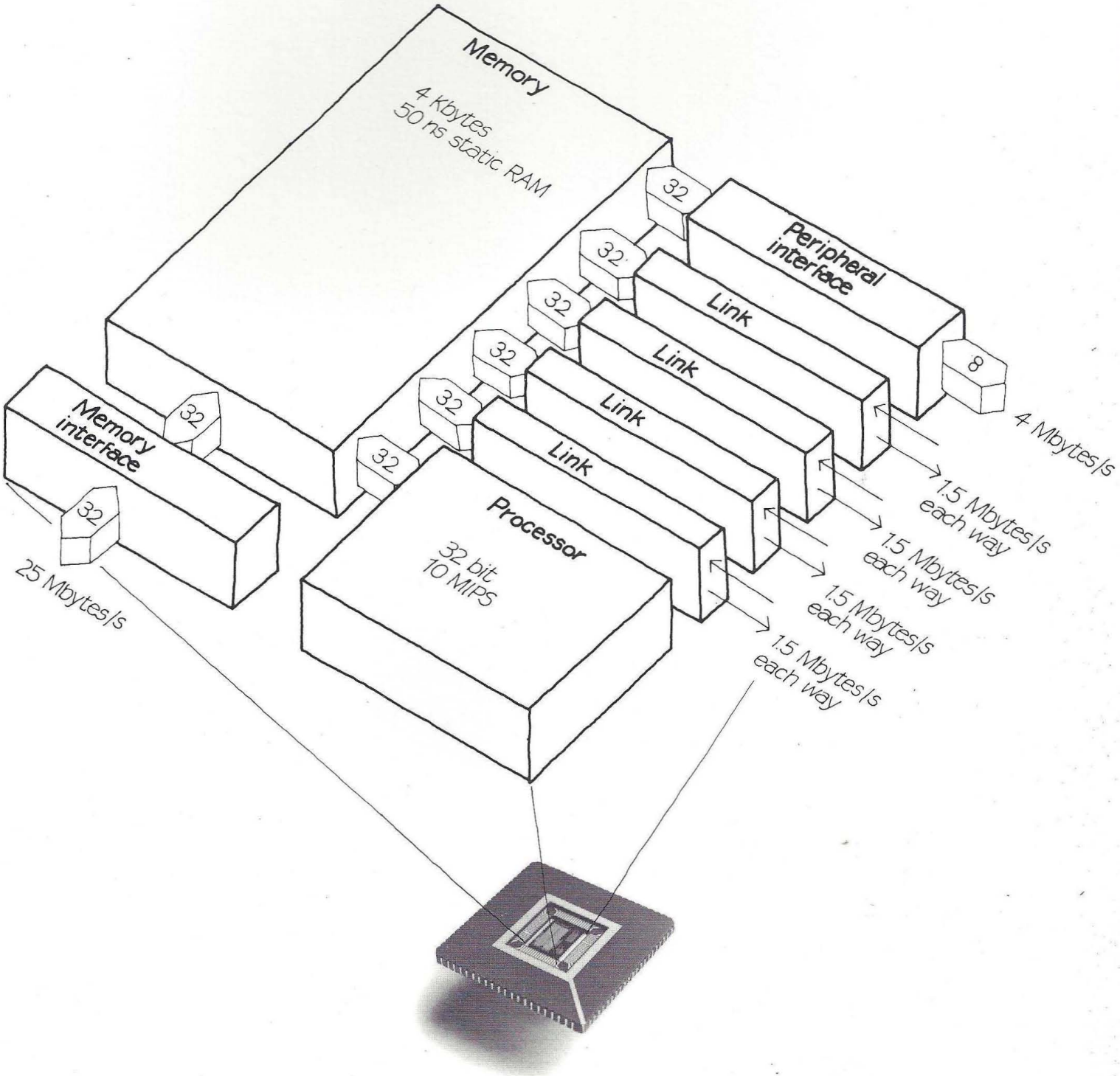
inmos

Advance information



IMS T424 transputer


Memory
4 Kbytes
50 ns static RAM
Memory interface
Processor
32 bit
10 MIPS
Peripheral interface
Link
Link
Link
Link
32
32
32
32
32
32
32
8
25 Mbytes/s
4 Mbytes/s
1.5 Mbytes/s each way
1.5 Mbytes/s each way
1.5 Mbytes/s each way
1.5 Mbytes/s each way

IMS T424	32 bit system providing 10 MIPS (millions of instructions per second) processing power with memory and concurrent communication capability, all on a single chip.
Processor	Reduced instruction set for compact programs, efficient high level language implementation and direct support of concurrency. High performance arithmetic with 50 ns basic instructions, 600 ns process switch and 950 ns multiplication.
Memory	4 Kbytes of static memory giving a maximum data rate of 80 Mbytes/s. Multiport access for processor, peripheral interface and each INMOS link.
Memory interface	32 bit multiplexed interface, with programmable timing to support mixed memory systems. Extends direct address memory space up to 4 Gbytes with a maximum data rate of 25 Mbytes/s.
Peripheral interface	8 bit bi-directional interface, providing connection to industry standard devices like controllers, processors and memory. Concurrent block transfer capability of 4 Mbytes/s.
INMOS links	Four INMOS standard serial links providing concurrent message passing capability to other transputer devices. Programmable data rate up to 1.5 Mbytes/s full duplex on each link to enable local and remote connection.
Technology	250 000 devices fabricated in an advanced 2 micron cMOS process with polycide interconnect. Laser fuse redundancy for improved economics and availability. 45 mm^2 chip dissipating less than 0.9 W, mounted in an 84 contact leadless chip carrier.
Engineering	5 MHz clock for simple engineering of transputer systems. Separately optimised interfaces for memory, peripherals and transputers ensure high performance with minimal glue.
Programming	Programmable in most standard high level languages: Ada, Basic, C, ... Direct execution of occam for maximum performance and exploitation of concurrency. Interactive program development using occam as the lowest level image of the system.
Range	Total compatibility at program and interface level with all transputer products, current and future. Full range, including 16 bit transputer, disk and graphics processors, development systems and software.

The transputer is a programmable component.

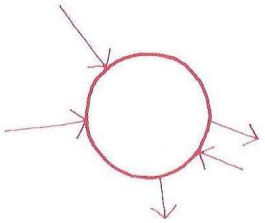
It is designed to exploit the opportunity of VLSI, providing performance and convenience through simplicity. Its revolutionary architecture enables the potential of concurrency to be realised for the first time, making today's applications easier to implement and creating a new dimension for tomorrow's systems.

The transputer uses silicon capability to make programming simpler and to make engineering easier than for any previous microprocessor.

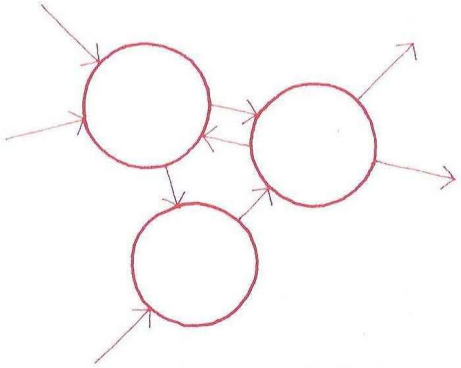
The architecture has been optimised to obtain the maximum of functionality for the minimum of silicon. It allows different trade offs between performance and cost, always giving an intrinsic advantage over older architectures.

The architecture is future-proof. It spans the range of applications from microcontrollers to supercomputers. Transputers will exploit future levels of integration by increasing the amount of processing, memory, communications and concurrency within the same architecture.

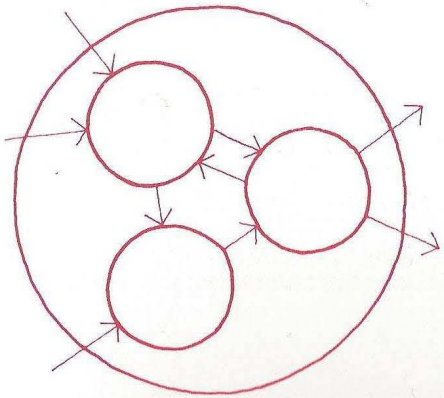
The occam calculus enables more powerful systems to be designed using interconnected transputers. Such concurrent systems unlock the processing potential of VLSI for fifth generation applications and provide the basis for future silicon systems.



A process is a black box, with inputs and outputs, that can process information



Processes can be connected together by channels to build more complex concurrent systems



A collection of processes is itself a process, so that processes may have internal concurrency

The transputer provides a direct implementation of the process model of computing. A process is an independent computation, with its own program and data, which can communicate with other processes executing at the same time. Communication is by message passing, using explicitly defined channels.

The transputer is designed so that it can implement a set of concurrent processes. The processes are time shared by the transputer, and special instructions are provided to support the process model of communication.

In addition, the transputer is designed so that its external behaviour corresponds to the formal model of a process. As a consequence, it is possible to program systems containing multiple interconnected transputers in which each transputer implements a set of processes. Since a program is defined as a set of processes, it can be mapped onto such a system in a variety of ways, for example to minimise cost, or to optimise throughput, or to maximise the responsiveness to specific events.

The transputer can be programmed in most standard programming languages, but its special properties are best exploited by programing in occam, the concurrent programming language developed by INMOS. The novelty of occam is in its treatment of concurrency. Occam provides the programmer with facilities for exploiting concurrency by encapsulating the notions of input, output and interrupts within a simple formalism. Occam may be used on its own, or as a harness linking modules written in other high level languages, thereby enabling them to be used concurrently.

The transputer architecture is optimised to execute occam. Compiled occam is as efficient as hand coding, so the need for an assembly language is eliminated.

An occam program may be executed by a network of transputers. However, the same program may be executed unchanged by a smaller network or even by a single transputer.

Occam enables a system to be described as a collection of concurrent processes, which communicate with each other and with peripheral devices through channels.

Occam programs are built from three primitive processes:

Assignment	Changes the value of a variable.
Input	Receives a value from a channel.
Output	Sends a value to a channel.

Occam encapsulates input, output, concurrent processes and interrupts in a simple, efficient language. It can be used in conventional microprocessor applications without the overheads of an operating system.

Processes are combined to form sequential, parallel or alternative constructs:

Sequence	The component processes are executed one after another.
Parallel	The component processes are executed together.
Alternative	The component which is ready first is executed.

Occam can capture the hierarchical structure of a system by allowing an interconnected set of processes to be regarded from the outside as a single process. At any level of detail, the designer is concerned with a manageable set of processes.

A construct is itself a process, and may therefore be used as a component of another construct.

Conventional sequential programs can be expressed in occam using variables and assignments, combined to form sequential constructs.

Concurrent programs can be expressed in occam using channels, inputs and outputs, which are combined to form parallel and alternative constructs.

Each channel provides a one way connection between two concurrent processes. Communication is synchronised and will occur when both processes are ready. The first process to be ready must, therefore, wait for the second process. When the second process becomes ready, the message is transferred and both processes can then proceed.

An alternative process may be ready for input from any one of a number of channels. In this case, the input is taken from the channel which is first used for output by another process.

The IMS T424 transputer is the first of a compatible range of products from INMOS.

The IMS T424 is a 32 bit transputer with an instruction rate of 10 MIPS. It consists of a small, fast processor, 4 Kbytes of memory, specialised interfaces to connect external memory and peripherals, and four INMOS standard links.

The IMS T424 transputer can respond to external interrupts within 600 ns typically; it can support simultaneous message (block) transfers between the peripheral interface, the four INMOS links and memory without a significant reduction in the processor performance.

The on-chip memory is 4 Kbytes of static RAM with a cycle time of 50 ns. The use of on-chip memory gives a significant advantage in memory data rate and power consumption. The memory can be extended off-chip up to a total of 4 Gbytes in a linear 32 bit address space. There is no difference, so far as the program is concerned, between on-chip and off-chip memory, apart from performance.

The use of serial links to implement the INMOS standard interface enables separate memory and peripheral interfaces to be provided without using an excessive number of contacts on the package. Each interface can, therefore, be optimised for its specific function, improving its performance and reducing the need for external glue logic.

The IMS T424 is implemented in an advanced 2 micron cMOS process. There is one layer of metal interconnect and one layer of silicide interconnect. A unique processing technique provides 1.75 micron transistors with low Miller capacitance, giving high performance.

The design has been carried out using the INMOS design system to ensure rapid implementation with high density layout. Equally important, occam has been used to provide a formal description which completely specifies the product, providing for the first time a rigorous environment for VLSI design.

The IMS T424 directly supports occam. Its behaviour is defined as an occam process and it executes an occam program. Each INMOS link provides two occam channels, one output and one input, for connection to other transputer products.

The IMS T424 process consists of a number of hardware processes representing the main concurrent elements of the transputer:

The processor executes an occam program received as a message through one of the transputer links.

The link controllers determine the state of their channel pairs and communicate with external transputer devices.

The memory interface controller determines the state of the memory interface and communicates with external memory.

The peripheral interface controller determines the state of the peripheral interface and communicates with external industry standard devices.

The processor communicates with these hardware processes using a set of standard channels which can be controlled by program.

```
PROC T424 (CHAN Ext.Mem.Outputs, Ext.Mem.Inputs,
                Link.Outputs, Link.Inputs,
                Peri.Outputs, Peri.Inputs,
                Event.Input) =
  CHAN Peri.Out[4], Peri.In[4],
       Link.Out[4], Link.In[4],
       Link.Set[4],
       Ext.Mem.Set[4],
       Ports.To.Mem[6], Ports.From.Mem[6]:
  PAR
    Processor (Ports.To.Mem[5], Ports.From.Mem[5],
               Ext.Mem.Set,
               Link.Out, Link.In,
               Link.Set,
               Peri.Out, Peri.In,
               Event.Input)
    PAR n = [0 FOR 4]
      Link.Cont (Ports.To.Mem[n], Ports.From.Mem[n],
                 Link.Outputs[n], Link.Inputs[n],
                 Link.In[n], Link.Out[n],
                 Link.Set[n])
    Mem.Int.Cont (Ports.To.Mem, Ports.From.Mem,
                  Ext.Mem.Outputs, Ext.Mem.Inputs,
                  Ext.Mem.Set)
    Peri.Int.Cont (Ports.To.Mem[4], Ports.From.Mem[4],
                   Peri.Outputs, Peri.Inputs,
                   Peri.Out, Peri.In)
```

The processor is designed to execute high level languages efficiently and will normally be programmed using occam or an industry standard language such as C or Pascal.

Concurrent processing

The processor executes programs sequentially. It implements parallel processes by sharing its time between the set of processes which are active at any instant. A process is active when it is not waiting for input or output.

The process currently executing runs until it has to wait for communication. When this happens, the process is set inactive and the next process on the active queue starts to execute. When a communication is made to a channel which is ready, the message is passed, the waiting process is linked to the end of the active queue and the current process continues to execute.

Priority processes

The T424 supports two levels of priority – in occam notation, a PRI PAR (priority parallel) process may have two components. A queue of active processes is maintained for each priority level, and a priority 1 (low priority) process is executed whenever there are no active priority 0 (high priority) processes.

If there are no active priority 0 processes, the latency (that is, the time from an external channel becoming ready, to the start of the first instruction of the relevant waiting priority 0 process) is typically 600 ns, maximum 2600 ns. Otherwise, if a priority 0 process is already executing, the relevant waiting process is linked to the end of the priority 0 queue.

Performance

The size of a program is given by the sum of the sizes of its program elements.

All timings are averages and the maximum time to execute the program is given by the sum of the times to execute the individual program elements.

If the program is held in external memory, the external program time must be added to obtain the program execution time. Because of the instruction lookahead and the overlap with internal memory this overhead will usually be small.

If data is held in external memory, the external data time must also be added to obtain the program execution time.

The processor shares memory cycles with the interfaces. Each concurrent access by an interface channel delays the processor by an average of 30 ns. The maximum reduction in performance is 10%; under typical conditions the reduction is negligible.

		Program size (bytes)	Execution time (ns)	External program access† (ns)	External data access† (ns)
Arithmetic operators	+, −	1	50	p	0
	* (multiplication)	1	950	0	0
	/ (division)	2	1950	0	0
	\ (remainder)	2	1950	0	0
Comparison operators	>, =, <>, <, <=, >=	2	100	2p	0
Logical operators	AND, OR	1	50	p	0
	/\, \/, >< (xor)	2	100	2p	0
Shift operators	<< [n], >> [n]	2	50n+50	0	0
Identifiers	variable	1.7	120	1.7p	d
	vector variable	2.7	160	2.7p	d
Expression evaluation	constant	1.3	70	1.3p	0
	parenthesis	0.5	50	0.5p	0.5d
Constructors	SEQ [n]	0	0	0	0
	PAR [n]	9n-7	450n-200	7np+nd	d+4nd
	ALT [n]	8n+7	600n+600	8p+2d	5d+4nd
	IF [n]	3n	150n	2nd	0
	WHILE	4	200	2d	0
Primitives	! (output), ? (input)	4	625	3p	5d
	! [n], ? [n] (vector)	4	50n+625	3p	5d+nd
	:= (assignment)	0	0	0	0
	:= [n] (vector)	4	100n+300	3p	2nd

†The additional time for access to external memory depends on the type of memory. The parameters for various types of INMOS memory are:

	p	d
IMS 1400-70 static RAM	20	100
IMS 2600-12 dynamic RAM	40	150
IMS 2600-15 dynamic RAM	55	200
IMS 3630-20 erasable ROM	55	200

One address instructions

Load local
Store local
Load local pointer
Load non-local
Store non-local
Load non-local pointer
Load constant
Add constant
Add to memory
Jump
Conditional jump
Call
Adjust workspace

Zero address instructions

Add
Subtract
Multiply
Divide
Remainder
Long add
Long subtract
Long multiply
Long divide
Normalise

Difference
Greater than
Equal zero

And
Or
Xor

Shift left
Shift right

Move message
Input message
Output message

Load byte
Store byte
Byte count
Word count
Byte subscript
Word subscript
Check subscript
Extend to word
Check partword
Extend to double
Check word
Read timer
Test error
Reverse
Return
Minimum integer
Initialise

Start process
End process
Alt start
Enable channel
Enable timer
Disable channel
Disable timer
Alt wait

The processor is optimised to implement occam. Its implementation is not an invariant of the transputer architecture, and user access to the instruction set is not supported. Future versions of the transputer will, therefore, be able to benefit from the greater capability offered by silicon, without obsoleting the user's investment in software.

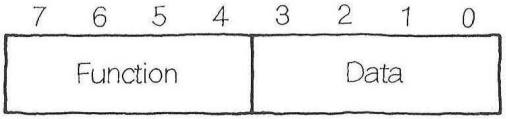
The processor is designed for performance and efficiency. This is achieved by designing the instruction set to simplify instruction decoding, by having the mimimum of special registers, by incorporating functionality into the registers and by the use of an evaluation stack. This approach simplifies compiler design, since all the operands are in a uniformly addressed data space, and also gives a fast process switch time.

The registers and functional units are regular bit slice designs with the data and control buses organised orthogonally. Control signals are generated from a microcode ROM which is optimised to match the control line pitch of the datapath.

The instruction set is compact. This is achieved by separating data access from data manipulation. Although the number of instructions is small, the functionality of the instruction set is high, so that typically programs require fewer instructions than on other microprocessors. As well as the conventional arithmetic and logical operations, the instruction set provides support for extended arithmetic, record access and specialised operations to support occam.

All instructions are one byte long and are divided into two four bit fields: function and operand. A technique is used whereby a sequence of instructions can be used to extend either field in units of four bits up to the word length, enabling both functions and operands to be frequency encoded.

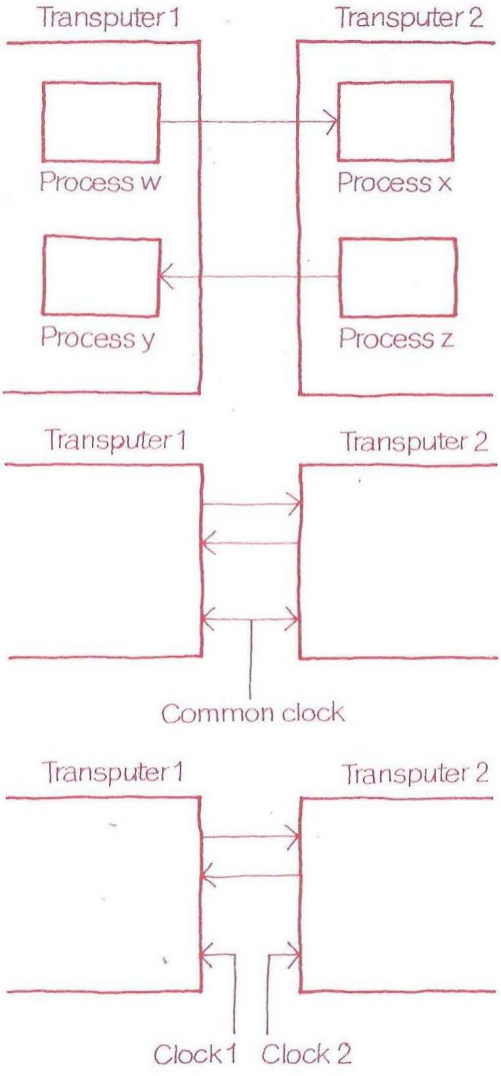
Instruction format



The instructions fall into two classes:

One address instructions where the operand is used by the function as a value;

Zero address instructions where the operand is used to define an operation on the values in the evaluation stack.



The IMS T424 has four standard INMOS links providing high speed intercommunication between transputer products, enabling a rich variety of networks to be constructed. Each link operates independently, and provides a memory to memory message (block) transfer capability between transputers.

Each link implements two occam channels. The link consists of an output and an input, both of which are used to carry data and link control information. A message is transmitted as a sequence of bytes. After transmitting a data byte, the sending transputer waits until an acknowledge has been received, signifying that the receiving transputer is ready to receive another byte. The receiving transputer can transmit an acknowledge as soon as it starts to receive a data byte so that transmission is normally continuous. This asynchronous protocol guarantees reliable transmission in spite of possible delays in either the sending or the receiving transputer.

During transmission of a message, both the sending and the receiving processes will be set inactive, and they will only be linked to the end of their respective active queues after the final byte has been acknowledged.

The data rate on each link can be set by program, using the LinkSet configuration channel. The highest frequency is 20 Mbits/s, giving a maximum data rate of 1.8 Mbytes/s on a channel.

INMOS links can readily be used for interconnection on a printed circuit board, or between boards on the same backplane. Because the link is TTL compatible its range may be extended by inserting standard (eg RS422/423) line drivers and receivers in each of its signal lines. The link is straightforward to use; provided the designer does not exceed the maximum capacitance specification, the only design requirement is to ensure that the input clock frequencies of two interconnected transputers are within specification.

Memory interface signals

Ras	→ Address valid
Cas	→ Column address strobe
Re	→ Read data enable
We0-3	→ Byte 0 to 3 write data valid
Amux	→ Address multiplex
AD0-31	↔ Address/data
Mcycle	← Multiplexed/non-multiplexed cycle select
Await	← Asynchronous wait input

The memory interface can support mixed memory systems, generating signals for both non-multiplexed and multiplexed memory, the memory timing being selectable by program. The timing can also be controlled externally by a wait signal.

The memory interface is a 32 bit multiplexed data and address bus. It extends the internal address capability to a total of 4 Gbytes in a single linear address space. The non-multiplexed cycle provides timing signals to drive industry standard static RAMs and ROMs. The multiplexed cycle provides timing signals for RAS and CAS and control for an external address multiplexer. The interface can also generate CAS before RAS refresh cycles.

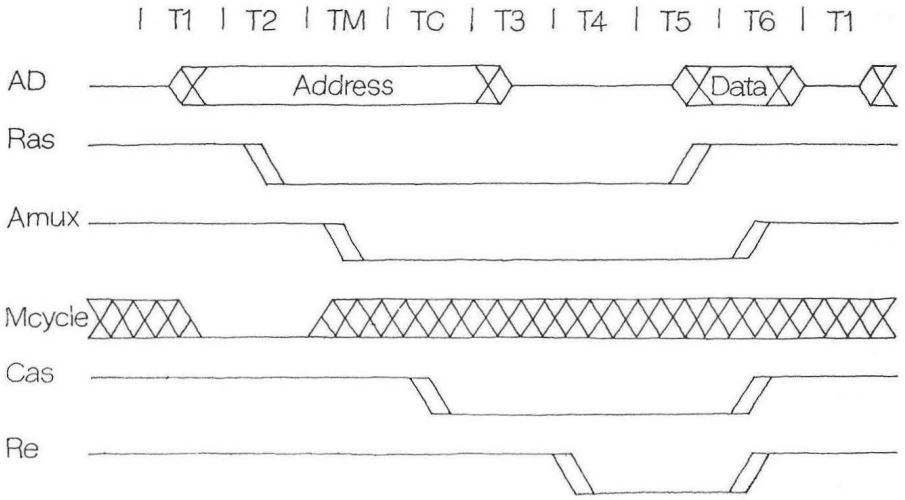
The type of memory cycle can be selected externally after the address has been output, enabling mixed memory systems. An asynchronous wait input is provided so that the memory timings can also be determined externally, if required.

The memory timings for the non-multiplexed and the multiplexed cycles and the refresh frequency can be set by program using the ExtMemSet configuration channel.

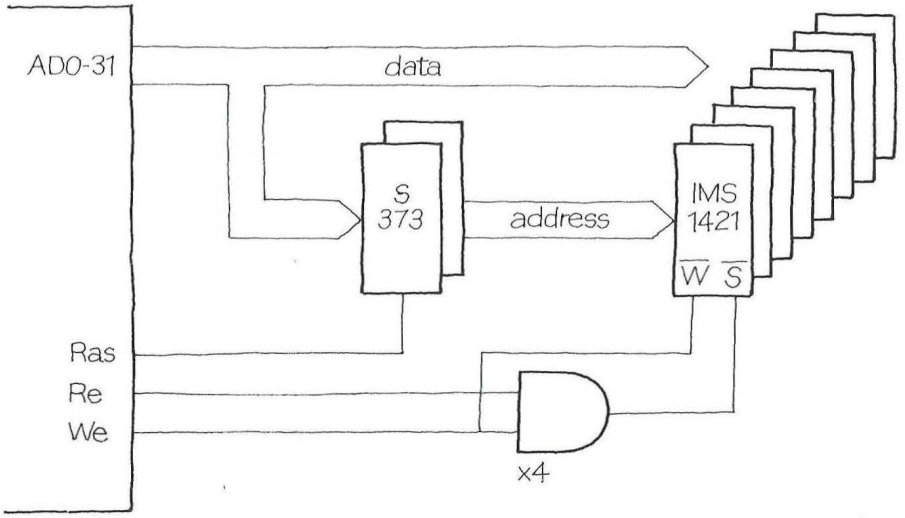
The timing diagram shows a read cycle for multiplexed address dynamic memory. The time for each phase is 25 ns, which gives a cycle time of 200 ns and a data rate of 20 Mbytes/s. For slower memories, the access and cycle times can be extended in increments of 50 ns.

Higher performance can be obtained using static memory. The logic diagram opposite shows the transputer connected to eight IMS 1421 4K x 4 static RAMs. The cycle to access static memory does not require the phases for address multiplexing and so is completed in 150 ns, giving a data rate of 25 Mbytes/s.

Multiplexed read cycle



Memory interface driving eight 4K x 4 static RAMs



Peripheral interface signals

Iclk	←	Peripheral interface clock
Req	→	Request for data transfer
Ack	←	Acknowledge data transfer
I/O	→	Direction of transfer
C0,1	→	Which of four channels is being addressed
D0-7	↔	8 bit wide data bus
Event	←	Event input to schedule a process

The peripheral interface provides access to industry standard devices such as controllers, memory and microprocessors. The interface controller provides a message (block) transfer capability between memory and the peripheral interface.

The peripheral interface is an 8 bit bidirectional bus which may be used to input or output sequences of bytes. There are two control lines which may be used to address external devices, and an Event input to provide an interrupt capability.

The interface is accessed via four standard output channels and four standard input channels. All eight channels use the same 8 bit path and transfers are handshaken, with the processor initiating the transfer. The transfers are synchronised to a separate external clock, which need not have any fixed relationship with the transputer input clock. Asynchronous operation is also permitted, but at a lower speed than for synchronous operation.

Externally addressable devices may be connected via the peripheral interface, for example, by using one output channel as the address channel, another as the write-data channel, and one input channel as the read-data channel. Both addresses and data may be arbitrarily long sequences of bytes.

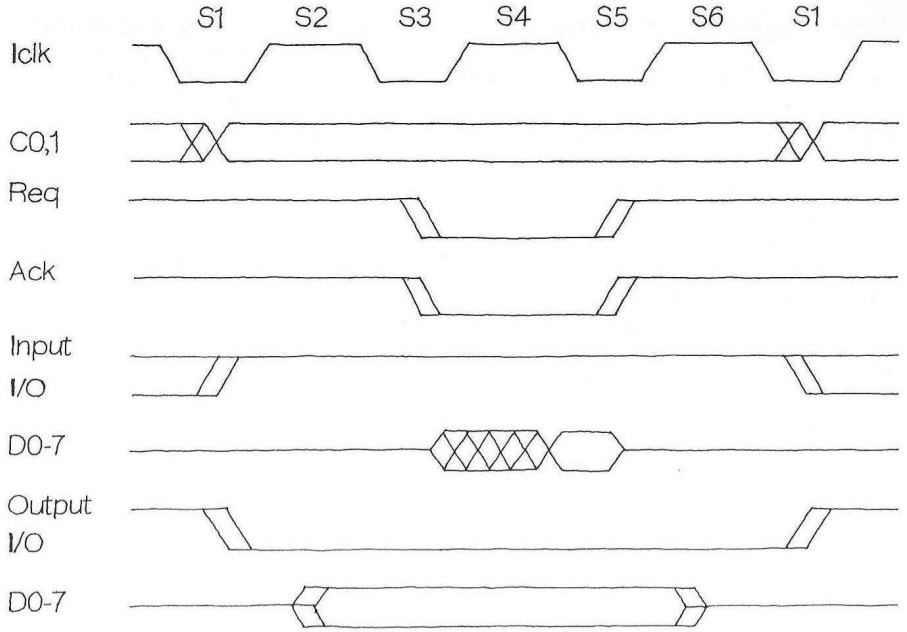
The 4 Mbytes/s data rate provided by the interface allows the connection of high performance peripheral chips, without the need for FIFOs or DMA controllers.

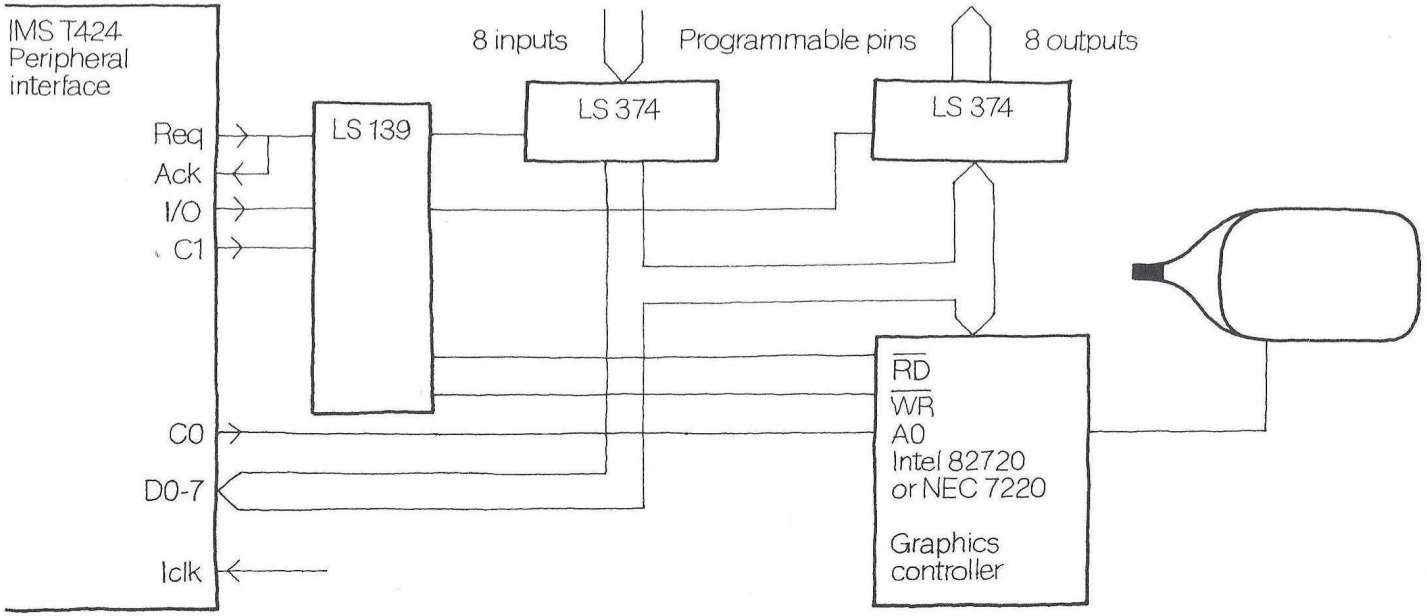
The Event input may be used to communicate with a waiting process, and hence cause it to be scheduled. This provides a similar function to an interrupt, in a manner consistent with the process model of the transputer. The typical latency for this interrupt is 600 ns. The Event input can also be used to enable the peripheral interface to respond to being accessed from a standard microprocessor bus.

The timing diagram shows how the signals provided by the interface are used.

The logic diagram shows how the interface can be connected to a graphics display controller, as well as to latches which can be used as programmable pins. The only glue logic required is a decoder.

Peripheral interface timing





The transputer is designed to simplify engineering and to ensure physical compatibility between all transputer products, current and future.

Clocking

A low frequency clock is used in order to simplify distribution of the clock; for compatibility, all transputers use the standard clock frequency of 5 MHz, irrespective of their internal performance.

Communication between transputers is independent of phase, so that there is no need to deskew the clock. Also, communication is not critically dependent on frequency, so that transputers with independent clocks can intercommunicate reliably.

Timer

The IMS T424 timer is a 32 bit integer. It is incremented every fifth cycle of the clock input, providing a time resolution of one microsecond.

Signature

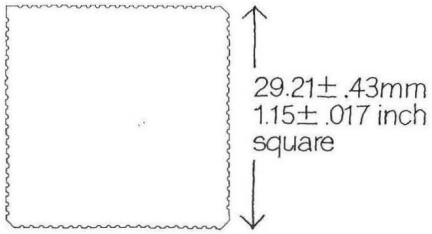
Each transputer is given a unique signature during manufacture. The signature, which includes the transputer device type, can be accessed by program through a standard channel.

Initialisation

After reset, and depending on the state of the Boot input, the transputer may either execute program from external ROM, or may bootstrap load itself down any of its links.

Packaging

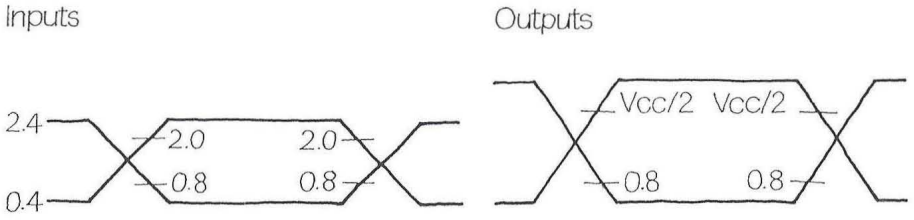
The IMS T424 is packaged in an 84 contact ceramic chip carrier.



DC characteristics

Parameter		Conditions	Min	Max	Unit
V_{IH}	High level input voltage		2.0	Vcc+0.5	V
V_{IL}	Low level input voltage		−0.5	0.8	V
I_I	Input current	0<=V_{IN}<=Vcc		±10	μA
V_{OH}	Data outputs Control outputs	I_{OH} = −4 mA I_{OH} = −6 mA	Vcc −1 Vcc −1		V V
V_{OL}	Data ouputs Control outputs	I_{OL} = 4 mA I_{OL} = 6 mA		0.4 0.4	V V
I_{OZ}	Output current in tri-state	0<=V_{OUT}<=Vcc		±10	μA
P_D	Power dissipation	f_{CLK} = 5 MHz		900	mW
C_{IN}	Input capacitance	f = 1 MHz		10	pF
C_{OZ}	Output capacitance in tri-state	f = 1 MHz		12	pF

Test waveforms and timing reference points



The transputer has been designed to simplify the task of programming and developing applications.

The transputer can be programmed in most high level languages, and its instruction set has been optimised to ensure that compiled programs will be efficient. Where it is required to exploit concurrency, but still to use standard languages, occam can be used as a harness to link modules written in the selected language.

To gain the most benefit from the transputer architecture, the system can be programmed in occam. This provides all the advantages of the best high level languages, the maximum program efficiency and the ability to use the special features of the transputer.

Program development can start now, using the occam programming system. As well as compiling programs for the transputer, this system is available with compilers for the VAX, the iAPX 86 family and the 68000 family. The system enables users to develop programs and to run them on existing microprocessor systems, prior to the availability of the transputer.

The occam programming system provides an ergonomic interactive environment for program development, with the ability to monitor the detailed behaviour of programs. The messages passing along any occam channel can be monitored in the same way as an engineer would monitor the signals passing between circuit devices, giving a new and powerful way of developing real time applications.

The facilities available include:

Interactive editing of programs.

Structured program display.

Integrated editing and compiling.

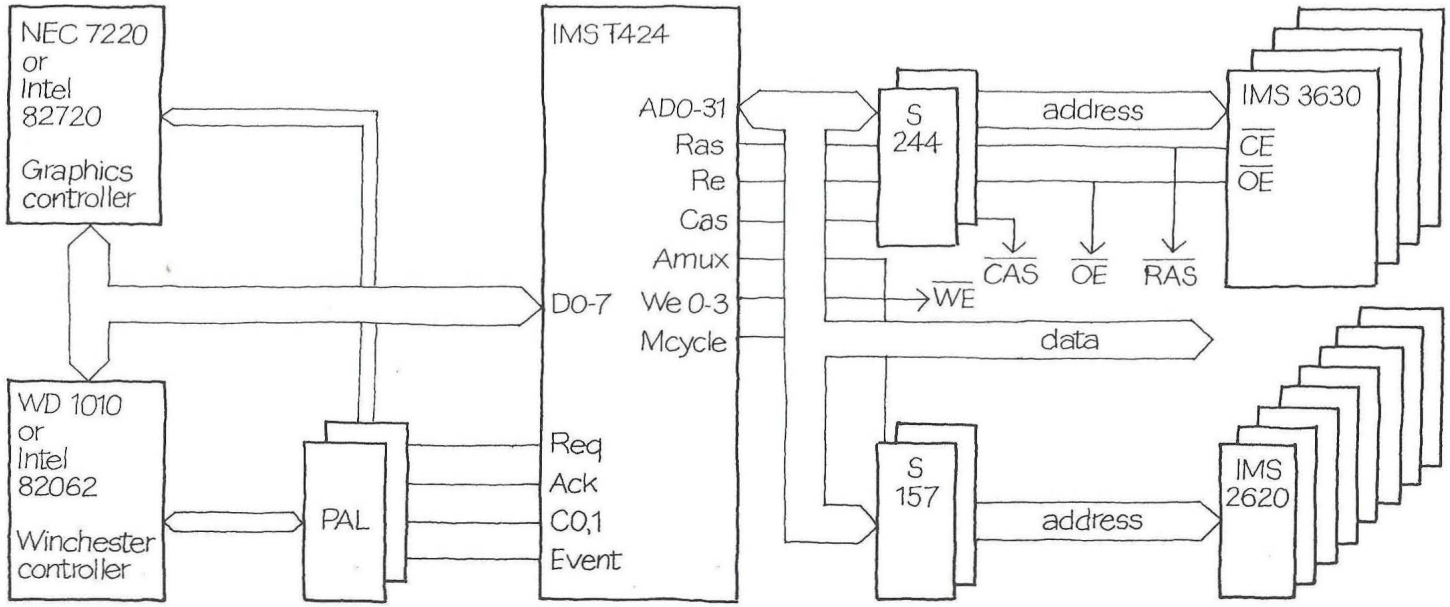
Monitoring and debugging at the occam level.

The transputer is a high performance component when used in single transputer configurations.

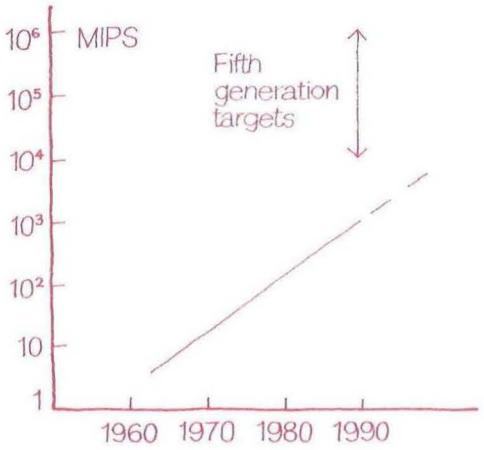
The separation of the peripheral interface from the memory interface enables both interfaces to be optimised for their function. The peripheral interface can support DMA from peripheral controllers, while the memory interface can support both EEPROM and dynamic RAMs with high performance and minimal glue.

This typical configuration can be the basis of a high performance personal computer or process control computer. The system software is held in 32 Kbytes of EEPROM, while the dynamic RAMs provide 64 Kbytes of user memory. The industry standard graphics display controller and Winchester disk controller connect to the peripheral interface with just two PALs – which act as decoders, address latches and tri-state buffers. The data rate of 4 Mbytes/s available from the peripheral interface means that no FIFOs are required to handle the 0.6 Mbytes/s data rate from the disk controller.

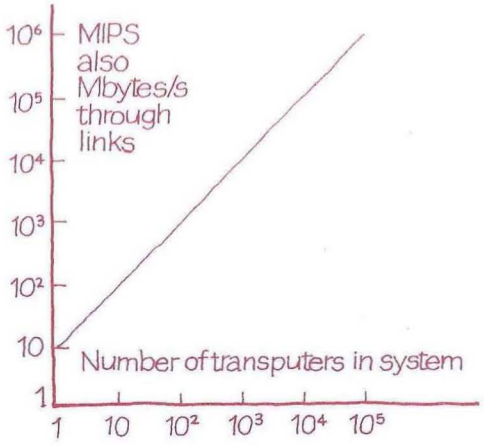
The system shown represents exceptional performance with low cost resulting from the small number of chips used in the system. Functionality can be increased by putting a local area network controller on the peripheral interface. The memory can easily be increased, either by using 256K RAMs or by using more 64K RAMs.



Conventional system throughput



Transputer system throughput



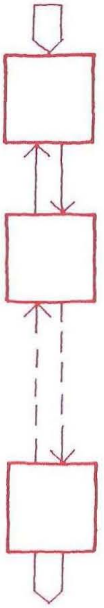
In the past, system performance has increased regularly by a factor of ten each decade. This improvement has been achieved by advances in circuit technology and by increasingly complex systems. For the future, VLSI offers the potential of much greater circuit complexity but only modest increases in circuit performance.

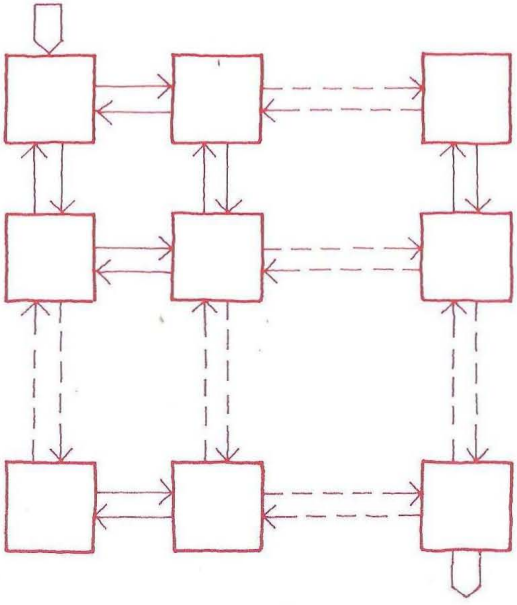
The economics of current systems are based on the historical perspective that processing is expensive in comparison with memory. This has led to the von Neumann bottleneck where a single processor is connected to vast amounts of memory. The economics of VLSI are different. Today, a single wafer of silicon can contain 2 Mbytes of memory or 256 conventional microprocessors.

To exploit this potential it will be necessary to build systems with a much higher degree of concurrency than is currently possible. The transputer is designed as a programmable component to implement such systems. The term 'transputer' reflects this new device's ability to be used as a system building block. The word is derived from 'transistor' and 'computer', since the transputer is both a computer on a chip and a silicon component like a transistor.

The power of the transputer is that it creates a new level of abstraction. Just as the use of logic gates and Boolean algebra provides the design methodology for present electronic systems, so the transputer, together with the formal rules of occam, provides the design methodology for future concurrent systems.

In their proposals to achieve intelligent interaction between people and computers, the Japanese have projected the need for computers with one thousand times the performance of present day systems. These will only be possible using concurrency, and the transputer has been designed to make such fifth generation systems a reality.





Pipelines and arrays of transputers can be used to provide greatly increased performance by exploiting the concurrency inherent in many applications. Two examples which require high performance are signal processing and database searching. Networks of transputers can provide the performance needed for both applications.

Signal processing, such as the fast fourier transform (FFT) algorithm, maps easily onto a pipeline. The pipeline can accept input samples at up to 100 kHz, which more than covers the full audio spectrum. A 64 point FFT requires six transputers in the pipeline, a 256 point FFT requires eight and a 1024 point FFT requires ten transputers. A pair of pipelines, interlinked at each stage, is able to accept input samples at up to 200 kHz. Higher frequencies can be handled by using more transputers in parallel.

A pipeline or an array can also be used for searching. Provided that the search requests can diffuse through the network, and the answers converge, the shape of the network does not matter – it can even contain faulty devices. The full internal memory of each transputer can be searched 1000 times per second. With external memory attached to each transputer the search rate is slower, but 64 Kbytes per transputer can be searched at least 30 times per second.

Other applications, such as image processing, finite element analysis (as used in weather forecasting), matrix manipulation, telephone switching systems, fault tolerant systems and artificial intelligence naturally lend themselves to arrays or networks of transputers.

Applications apparently less directly suitable have also been found to benefit from multi transputer systems. As an example of a compiler, an occam compiler, written in occam, can use a pipeline of processes. As an example of a large scientific program, the INMOS logic simulator can achieve performance proportional to the number of transputers used.

IMS T222 transputer

The IMS T222 is a 16 bit transputer. It has an identical instruction set to the IMS T424 and programs will behave identically on both the T222 and T424, providing the 16 bit range of the T222 is not exceeded.

The IMS T222 processor runs at 10 MIPS (millions of instructions per second), with 50 ns basic instructions, 600 ns process switch, and 950 ns multiplication.

The memory is 4 Kbytes of static RAM, giving a maximum data rate of 40 Mbytes/s, with multiport access for the processor, peripheral interface and each INMOS link. The memory interface is a 16 bit multiplexed interface with programmable timing, an address space of 64 Kbytes and maximum data rate of 13 Mbytes/s. The peripheral interface and the four INMOS links are identical to those on the IMS T424.

The IMS T222 is packaged in a 68 contact ceramic chip carrier.

IMS T424 transputer

The IMS T424 is a 32 bit transputer providing 10 MIPS, with 4 Kbytes of static RAM, a 32 bit multiplexed memory interface, an 8 bit peripheral interface and four standard INMOS links.

IMS G213 graphics processor

The IMS G213 is a high performance programmable graphics controller, suitable for high quality colour graphics. It includes a 16 bit processor, 2 Kbytes of static RAM and two INMOS links for connecting to transputer systems.

IMS M212 disk processor

The IMS M212 is an integrated programmable disk controller for floppy disks and Winchesters. It includes software configurable crc/ecc, as well as a 16 bit processor, 2 Kbytes of static RAM and two INMOS links for connecting to transputer systems.

Occam programming system

A personal workstation with 256 Kbytes of memory, twin floppy disks and a bit mapped screen, providing an initial programming tool for developing occam programs. Also available is the occam evaluation kit, an electronic tutorial system to demonstrate the basic principles of occam.

Future products

The IMS T424 is the first of a range of transputer products which will span the entire spectrum of microprocessor applications, from low cost microcontrollers up to supercomputers and fifth generation systems. All transputers will be totally compatible, whatever their word length, instruction set, processor speed, or interfaces.

Static RAMs

INMOS has a dominant position in the high performance 16K static RAM market and provides a complete range of products. These are used in mini and mainframe computers for cache, microcode store, memory management and fast buffers. Other applications include video and radar systems, instrumentation and telecommunications. Versions with access times down to 35 ns are readily available in plastic DIPs, ceramic DIPs and ceramic chip carriers. These products feature an advanced 2.5 micron single polysilicon nMOS process and use redundancy to give improved economics and availability.

IMS 1400 16Kx1 35/45/55/70/100 ns
IMS 1420 4Kx4 45/55/70/100 ns
IMS 1421 4Kx4 45/55/70/100 ns

Dynamic RAMs

INMOS introduced the world's first 100 ns 64K dynamic RAM in 1982. Since then, INMOS has introduced two further products, a 16Kx4 and an 8Kx8 configuration, making the widest rage of dynamic RAMs on the market. These are used in a variety of high performance applications from mainframe computer memory to video displays and peripheral buffers. These products feature an advanced 2.5 micron double polysilicon process and also use redundancy.

IMS 2600 64Kx1 100/120/150 ns
IMS 2620 16Kx4 100/120/150 ns
IMS 2630 8Kx8 120/150/200 ns

Erasable ROMs

INMOS released details of its IMS 3630 at ISSCC in February 1983. This is an 8Kx8 electrically erasable and programmable ROM with a single 5 V supply. It is particularly easy to program – the complete memory can be programmed in 1.3 seconds – and during this time, the programming device can perform other operations concurrently. The IMS 3630 is available in ceramic DIPs; it is fabricated using a novel 3 micron Nitrox process.

IMS 3630 8Kx8 200 ns

Military products

INMOS memory products are manufactured to the highest possible standards and are available to military specification in a variety of packages.

Future products

INMOS has under development a range of new memory products, providing improvements in performance and capacity. These products will use cMOS processes compatible with the transputer process.

INMOS was started in 1978 with the aim of becoming a world leader in VLSI. The strategy has been to use innovative design and advanced processing techniques to create products offering significant advantages to the user.

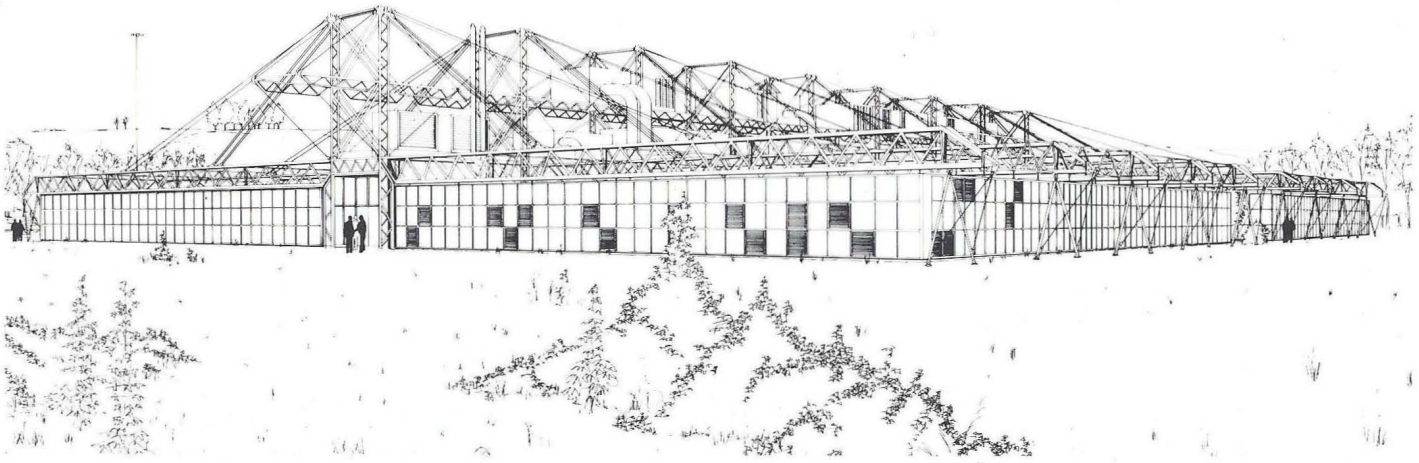
From the beginning, INMOS has operated internationally, benefiting from the strengths of both the UK and the US. The UK design centre in Bristol develops VLSI design techniques and microprocessor products. The US design center in Colorado Springs develops process technology and memory products.

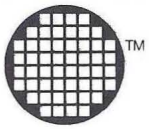
INMOS operates advanced manufacturing facilities in both the UK and US, using direct step on wafer lithography, ion implantation and plasma etch for all stages of manufacture.

The first stage of the INMOS strategy was to enter the memory market. This has been successfully achieved and INMOS has gained an outstanding reputation for the technical excellence of its static and dynamic RAM products.

The introduction of the transputer signals the start of the second stage of the INMOS strategy, which is to establish the company as a force in the microprocessor market.

Both memory and microprocessor capabilities are seen as precursors to the ultimate goal of INMOS to become a world leader in VLSI capable of delivering complete silicon systems.





Please add my name to your mailing list for:

☐ occam ☐ transputer ☐ memories

Name Title

Company

Address

Country Zip/Post code

Telephone

Your interest

Establishment size

Please mail to one of the following addresses for immediate attention

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA

INMOS GmbH
Untere Haupstrasse 7
8057 Eching
West Germany

INMOS Limited
Whitefriars
Lewins Mead
Bristol BS1 2NP
UK

Matsushita Electric Trading Co Ltd
Component Section
Import Division
26th Floor
World Trade Centre
4-1, 2-Chome Hanamatsu-cho
Minato-KU
Tokyo 105
Japan

INMOS Corporation
PO BOX 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000

INMOS Corporation
23505 So. Crenshaw Blvd
Suite 201
Torrance
CA 90505
USA
Telephone (213) 530 7764

INMOS Corporation
1735 North 1st Street
Suite 303
San Jose
CA 95112
USA
Telephone (408) 298 1786

INMOS Corporation
60 Mall Road
Executive Place V
Suite 207
Burlington
MA 01803
USA
Telephone (617) 273 5150

INMOS Corporation
5457 Twin Knolls Road
Suite 201
Columbia
MD 21045
USA
Telephone (301) 997 2947

INMOS Corporation
5200 West 73rd Street
Minneapolis
MN 55435
USA
Telephone (612) 831 5626

INMOS Limited
Whitefriars
Lewins Mead
Bristol BS1 2NP
UK
Telephone (0272) 290861

INMOS SARL
Immeuble Monaco
7 rue le Corbusier
SILIC 219
94518 Rungis Cedex
France

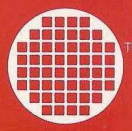
INMOS GmbH
Untere Hauptstrasse 7
8057 Eching
West Germany
Telephone (089) 319 10 28

Designed by HSAG
Printed in England by Syon Print Limited

November 1983

72-TRN-001 000

inmos™



INMOS Limited
Whitefriars
Lewins Mead
Bristol BS1 2NP
UK
Telephone (0272) 290861
Telex 444723

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000
Telex 910 920 4904

F12513FT