Concurrent Interconnect for Parallel Systems

PETER THOMPSON
INMOS Ltd, 1000 Aztec West, Almondsbury, Bristol BS12 4SQ, UK

A new type of computer systems interconnect is motivated and described, which allows any number of devices to interact concurrently, scales well with system size and has a low implementation cost. The architecture of a commercially available device from which such interconnect can be constructed (the IMS C104) is described and an example of using such interconnect for communication between processors is given.

Received May 1993; revised September 1993

1. REQUIREMENTS FOR A NEW INTERCONNECT

Practically all modern electronic systems are 'parallel', in that they consist of a number of relatively complex subsystems connected together. Even in a humble PC there are subsystems for I/O, for controlling the keyboard and screen, interfacing to networks and so forth. At the other end of the scale, it is increasingly widely recognised that the most economic way to build very high-performance systems is by using many identical components of moderate performance working together. Such systems can provide very high computational power [in massively parallel processors (MPPs) such as the CM-5 or the Parsytec GC-machine], fast response (for transaction processing or distributed control), and very large I/O throughput [as in telecommunication switches and redundant arrays of inexpensive disks (RAID) systems]. They also have the potential to be more reliable, maintainable and expandable than conventional, monolithic systems.

The construction of high-performance systems with parallel processing and/or parallel I/O demands a fast, cheap, scalable, low-latency interconnect. The interconnect must be fast and low-latency, otherwise it will be the limiting factor in system performance, and it must be cheap, or else it will dominate the system cost. It must also scale well in both performance and cost relative to the system size, otherwise large parallel systems will either be limited in performance or highly expensive, perhaps both. Available interconnects do not meet these criteria, either because they are based on a bus, so that the contention for shared resources limits overall performance and scalability; or because they are designed for communication over long distances, which incurs high costs; or because they aim at the extreme of currently achievable performance, which increases costs out of proportion to performance.

1.1. Interfacing to the interconnect

The requirement of low cost implies that at least an *interface* to the interconnect must be implementable with a relatively small amount of circuitry in a non-exotic technology, since every subsystem must have such an interface. This means that the protocols used to communicate across the interconnect must be simple and require only minimal buffering, otherwise an unacceptable amount of circuitry will be required to implement them. An interface requiring even one extra chip would be considered expensive in high-volume applications, so the interface should ideally be integrated with another device, such as a processor or ASIC. Too many pins per connection would make such integration impractical, limit the maximum number of connections available on a device (or force the use of exotic, expensive packaging) and lead to skew-control problems, thereby increasing the cost of printed circuit boards (PCBs), etc. Thus, where 'cost' includes not only the price of components, but also the engineering effort required to use them successfully, each interface must require both a minimal amount of circuitry and a minimum number of wires.

1.2. Interconnect components

To ensure cost-effectiveness, any active components of the interconnect in addition to the interfaces should ideally be made with the lowest-cost technology, currently CMOS. Since there is an element of the system cost associated with each component which is independent of how large or complex the component is (including the cost of maintaining an inventory of that component, the cost of pick-and-placing it onto a circuit board, and so forth), it would be as well for the number of additional components to be small. Since some complexity is to be expected in a high-performance interconnect, this implies that the active components must be highly integrated. Putting these factors together suggests that the interconnect should be constructed using CMOS VLSI chips. Since modern CMOS processes provide a very high level of interconnectivity on chip (for instance an entire 64-bit bus can be fitted into the width of a single 0.15 mm PCB track) this seems to be a promising approach.

In order for VLSI chips to be cheap, they must be manufactured in large volumes, which means that they should not be too specialized in their application. For interconnect chips, this implies that few assumptions

should be made about the parallel systems in which they are to be used. Thus flexibility becomes another factor to be considered along with performance and chip size, since the economics of VLSI make it worthwhile to increase the size and/or complexity of a chip if this results in a significant increase in its applicability. Moreover, it will not be generally attractive to integrate an interface to an interconnect into other chips unless the interconnect is standard, which again implies that its applicability must be broad. Scalability is also important here, since an interconnect will not be widely used unless it is suitable for small systems as well as large ones. Systems requiring the interconnection of hundreds of subsystems will probably be unusual until after the turn of the century, so an interconnect which is only useful or cost-effective at such scales will not become standard.

Thus we see a requirement for an interconnect for parallel systems where the emphasis is (unusually) on cost-effectiveness, flexibility and applicability to small systems rather than maximum performance of the largest possible configuration. One approach to the construction of such an interconnect which has resulted in a commercially available CMOS device is outlined in the rest of this paper.

2. CONVERGING SOLUTIONS

In this section we examine current trends at the two extremes of the interconnect scale (local busses and telecommunications) to see whether they are showing any tendency towards meeting the requirements we have set out for a new type of interconnect.

For a relatively small number of components which are physically close together the standard method of interconnection is a bus and in many cases this is perfectly adequate. However, as the number of components and/or the distances between them increase a bus suffers from capacitive loading and contention. The fundamental limits now affecting bus performance are driving a trend to deep pipelining leading to a more 'packetized' type of interaction and also to increasing numbers of independent paths (Gustavson, 19??). Recognising that a cabled bus with a total length of cable up to several tens of metres would be severely limited by the bus capacitance, the P1394 Serial Bus (Teener, 1992) uses point-to-point cables and performs the actual bus function in silicon, where the lower capacitance offers an order of magnitude faster performance. Unfortunately *any* type of bus still suffers from the problem of increasing contention as the number of subsystems connected to it increases and so cannot provide scalable performance (Walker, 1992). Busses are also inherently inefficient, since the maximum average utilization of each subsystems' transmitting logic is inversely proportional to the number of subsystems on the bus.

The well-established trend in the electronics industry is for increasing levels of integration in individual subsystems and ever-increasing clock rates. Since the speed of light unfortunately does not increase, communication between subsystems tends to take a larger number of clock cycles. This can be seen in the trend towards processors whose internal clock speed is several times the frequency at which the bus operates. As the clock rate increases the 'distance' between subsystems (as a number of clock cycles) also increases, so todays' parallel system tends to resemble yesterdays' *distributed* system. Thus we may gain some insight into the parallel systems interconnect of tomorrow by looking at techniques used in telecommunications today.

Telecommunications systems are characterized by sparse connections, whose bandwidth must be shared between a number of uses. In the days when computational resources were expensive, it was easiest to allocate the bandwidth of a telecommunications connection in a static, predefined way such as time-division multiplexing (TDM). However cheaper processing has made it more cost-effective to allocate bandwidth dynamically by means of packet-switching. The most widespread form of packet switching is X.25 (Black, 1989), whose protocols contain a high overhead for error correction because of their origins in the days when typical bit-error rates (BERs) were 10^{-3}. Now that the use of optical fibres has brought BERs down to 10^{-8}, X.25 is being replaced with much simpler protocols such as Frame Relay (Dettmer, 1992), in which any error correction required is performed end-to-end.

Thus as busses become longer, faster or connect to more components, they are becoming serial, like telecommunications. On the other hand, as telecommunication channels become more reliable (like the channels which exist inside electronic equipment) and higher levels of integration make processing cheaper, telecommunications systems are moving towards lightweight packet switching protocols. Thus there is a convergence towards serial channels carrying ultra-lightweight packet protocols, with packet switching performed using the tremendous interconnective capabilities of VLSI. This becomes our starting point for a standard interconnect for parallel systems.

3. CONCURRENT INTERCONNECT

In this section we consider a consistent approach to constructing a parallel systems interconnect, following directly from the requirements and trends identified in previous sections.

As discussed previously, an interconnect for parallel systems must have high performance, good scalability and low cost. Since these requirements are no different from those of the whole parallel system, a similar approach can be followed in order to satisfy them. Just as a parallel system can provide high performance by allowing many identical components to work concurrently, the interconnect can provide a high aggregate bandwidth by allowing many separate connections with

moderate performance to operate simultaneously. This is exactly opposite to the operation of a bus, in which one high performance connection is used sequentially for each transfer. For both the parallel system and the concurrent interconnect, each component or connection need not be very high performance (provided it can be well-utilized), allowing both component and engineering costs to be kept down while providing high performance.

Busses are complex and inefficient because of the need to share access to the wires. When using a number of concurrent connections, maximum simplicity, modularity and fault-tolerance can be achieved by making each connection unshared, i.e. point-to-point. The number of wires per connection, which affects the cost of the interface, is minimized by making each connection serial. Thus we propose a concurrent interconnect consisting of many serial point-to-point connections operating simultaneously. This resembles a telecommunications network in miniature, but uses much simpler protocols to take advantage of the high reliability of connections within a system.

3.1. Serial routing

To connect many devices together using point-to-point connections, it is not feasible to provide a direct, physical path between every pair, even when the connections use a minimum number of wires. Nor is it acceptable to connect only certain pairs of devices unless the connection pattern happens to match that required by the application. Thus data must be able to travel from one node to another via one or more intermediate nodes, i.e. it must be through-routed. Although networks exist which allow any set of pairs of devices to be interconnected (Clos, 1953), in order for *every* pair of devices to be able to communicate, data must be routed in different ways at different times. One way of achieving this is to configure the intermediate nodes to make each connection before sending the data, such as in a telephone exchange. Unfortunately this generally requires that the destination of the data must be supplied in advance to a central controller, which increases latency and creates a system bottleneck unless the rate at which connections are altered is very low. This approach can be used in very tightly coupled, SIMD-style machines, in which the communication pattern is pre-determined, but since we wish to avoid restricting the applicability of our interconnect to a limited class of systems, a better solution is to make the data *self-routing* so that it contains within it the information which determines which way it should be routed. Self-routing interconnect is equally appropriate both in systems where data moves in an orderly fashion along pre-defined routes and in those whose communication requirements arise dynamically at run-time. It also enables the control function to be distributed, and hence to scale in performance with the size of the system.

We call a piece of data together with its associated routing information a *packet*. The requirement of low-latency implies that packets must be limited in length, since otherwise connections could be occupied indefinitely by long packets, making latency unbounded for other packets requiring the same connection. In addition, short messages are frequently required in parallel systems for control and synchronisation. Thus it is important to transmit such short messages efficiently, and so it must be possible to send them as packets which are no larger than absolutely necessary, and the overhead on each packet must be small. On the other hand, when sending a large amount of data it is more efficient to divide it into packets of a reasonable size in order to reduce the proportionate cost of the overheads. Thus the packet size must be variable to allow both large and small amounts of data to be sent efficiently.

3.1.1. *Wormhole routing*

A technique used for minimizing latency in parallel computers is *wormhole routing* (Ni and McKinley, 1993), in which only the header of the packet is initially read in by the routing node. The routing decision is taken, the header is output and the rest of the packet is sent directly from the input to the output without being stored in the node. This means that a packet can be passing through several nodes at the same time and the header of the packet may be received by the destination even before the whole packet has been transmitted by the source. Thus this method can be thought of as a form of dynamic circuit switching, in which the header of the packet, in passing through a sequence of nodes, creates a temporary circuit (the 'wormhole') through which the data flows. As the tail of the packet is pulled through, the circuit vanishes. The transmission of a single packet may thus be pipelined through a series of devices. As well as minimizing latency, wormhole routing has the further advantage that it is independent of the packet length, thereby making the interconnect more general purpose.

In a wormhole routing system, if a packet is unable to proceed because the output it requires from a node is busy, the progress of the packet must be stalled until the required output is free. This is usually achieved by adding extra handshaking signals to the physical channel, but this conflicts with the goal of minimising the number of wires per connection. An attractive solution to this dilemma is to use bi-directional connections, and use each set of wires to transmit both data and flow-control information. This trades a slight increase in protocol complexity for the elimination of wires which do not directly contribute to the data rate.

Wormhole routing systems typically consider each packet to be a multiple of a basic unit of data, called a 'flit', and control the flow of the packet by either allowing or preventing the transmission of each individual flit from one node to the next. Where wide channels are used, with extra wires to perform the flow-control, the

natural size of a flit is the width of the channel, but this approach breaks down when the channel is serial, since single bit flits are clearly not efficient. To minimize the overhead of multiplexing flow control information with the data, the flow of data must be managed in blocks of a reasonable size, but to consider such blocks 'flits' leads to a loss of efficiency when sending messages which are not a multiple of the block and implicitly imposes a minimum packet size, conflicting with the need to send small messages efficiently. A better solution is to decouple the flow control from the packets, so that a packet is regarded as a sequence of units which are as small as possible (e.g. bytes), whose flow is controlled at a coarser grain at the expense of a small amount of buffering. (Although Dally (1990) discusses a more general relationship between flits and *phits*, the physical unit of data transfer, he does not consider the possibility of a packet which is not an integral number of flits.) This results in a simple layered protocol, whose lower levels implement a bi-directional pair of handshaken FIFOs between each connected pair of devices. Packets then pass through these FIFOs, which are dynamically interconnected to perform routing.

3.1.2. *Advantages and disadvantages of serial connections*

Guided by converging trends, we have arrived at the combination of wormhole routing with lightweight layered protocols on serial channels. This is unusual; most, if not all, wormhole routing systems to date have employed considerable wider channels. One reason is that wider channels are easier to construct; a bi-directional data rate of 20 MBytes/s requires only 10 MHz signals on a pair of byte-wide channels whereas it requires 50 MHz signals on a four-wire bi-directional serial link (May *et al.*, 1993). The ability to implement such links reliably with low-cost technology has only been developed in the last few years. Another reason for the preponderance of wide channels is that the analyses performed by the parallel processing community favour low-dimensional networks. For instance, Dally (1990) concludes that two- or three-dimensional networks provide the best performance, which implies that routing nodes should have a valency (number of channels) of no more than six, each of which should be as wide as possible in order to maximize the performance of the routing node. However, this analysis is based on the assumption that the principal limiting factor is the number of wires crossing a bisection of the system. In considering the pin-count of the routing devices as another limiting factor, Dally (1991) concludes that somewhat higher-dimensional structures give better performance, as does Agarwal (1991) when considering the effect of node delay. However, none of these analyses consider the issue of system cost: they are only concerned with maximizing performance under one constraint or another. Using low-dimensional networks increases the minimum number of routing chips which are needed to interconnect a given number of devices and using wide channels increases the minimum number of pins which are required for the interface on each device. Serial channels provide the lowest minimum costs, e.g. using four-wire bi-directional serial links, a 32-valent single-chip router (described in Section 4) can be constructed in standard CMOS VLSI. This allows up to 32 devices to be connected full-duplex with only one routing chip and (for example) 512 devices to be interconnected using only 48 routing chips (May *et al.*, 1993). Lower valency routers can obviously be built at even lower cost. Serial channels also provide the maximum flexibility, since they can always be run in bundles to provide the bandwidth of a wider channel. A further advantage of the high valency available by using serial channels is the ease with which fault-tolerant interconnects can be constructed (Thompson, 1993).

The chief disadvantage of serial channels is of course the latency of packet transmission. While the overall bandwidth on a collection of serial channels can be as high as if the same number of wires were used for a smaller number of wider channels (or even higher, since the problem of skew between different signals is minimized), the latency for each packet is increased. The latency for the *header* of the packet to be delivered is not increased substantially provided the header can be kept short, but the time for the whole packet to be transmitted is increased, particularly if the packet is long, because its body is transferred at a lower rate. However, in message-passing systems it is frequently the latency for the arrival of the header which is crucial and, in distributed shared memory systems, packets are typically short, so the effect of the increase in latency is not so great as might at first be assumed.

There is clearly a trade-off between the number of genuinely concurrent communications supported by the interconnect and the maximum performance of an individual communication, and in this sense an interconnect based on serial channels is at the opposite end of the spectrum from a shared bus. There are many systems in which a bus is the most appropriate form of interconnect. However, as parallel systems become more prevalent, the number in which a concurrent interconnect is the most appropriate will increase. In many systems, both forms of interconnect may be used for different purposes, as discussed in Walker (1992).

4. COMPONENTS FOR CONCURRENT INTERCONNECT

We have developed a concurrent interconnect which can be constructed from standard VLSI components. In this section we describe the architecture of one such component, the IMS C104 packet routing chip.

The IMS C104 has 32 bi-directional serial interfaces. Each of these interfaces may be connected to a similar interface on another device (which may be another IMS C104) to form a bi-directional serial link, called a

DS-Link. (DS-Link is a trademark of INMOS Ltd.) Each DS-Link transmits at a programmable speed up to 100 MBits/s using a novel two-wire encoding which requires a maximum frequency of only 50 MHz on each wire. One wire carries the digital signal and the other wire (the 'strobe') changes state only when the data does not, so that only one wire is changing state at a given moment. [This is very similar to the self-timed signalling convention described in McAuley (1992).] These signals are decoded with self-timed logic which has only to discriminate the order in which edges occur on the two wires, giving a whole bit-time of skew tolerance. The self-timed decoding circuitry enables a DS-Link to receive data at any rate, regardless of its transmission speed. Each byte of data is transmitted as a 10-bit sequence, which includes a parity check bit and a flag to distinguish control sequences, which are used to delineate the ends of packets, and to implement the flow-control mechanism. The protocols used on DS-Links are described in detail in May *et al.*, (1993) and INMOS (1993), and are the subject of an IEEE draft standard (Whitby-Stevens, 1993).

The data arriving on each link is interpreted as a sequence of packets, each starting with a routing header and ending with a special control character called a terminator. Headers are kept short to minimize latency and the overhead on bandwidth, and may be 1 or 2 bytes. One byte headers are sufficient for systems of moderate size (up to 256 connections), while 2 byte headers allow very large systems to be constructed (up to 64 k connections). Each link is equipped with an independent packet processing engine which can route over 7 million packets per second, all of which operate concurrently, giving an aggregate peak processing rate of over 220 million packets per second. Each packet processor contains a programmable set of registers which define the routing algorithm, using the concept of interval routing (van Leeuwen and Tan, 1987). Even though in most cases the registers of each link will be programmed identically, so that the routing of a packet depends only on the value of its header, the provision of a set for each link ensures that there is no contention for shared resources within the device, and enables the device to be partitioned into separate logical devices for system security, if required.

The main functional components of the IMS C104 are illustrated in Figure 1. The IMS C104 implements wormhole routing, as previously described, so that the routing decision for each packet can be taken as soon as the header has been received. The data streams through the links are interconnected by a full 32-way crossbar inside the IMS C104, so that the only contention which occurs is when two or more packets are routed to the same output. To deal with this case, there is high-speed arbitration circuitry associated with each link output which allows one of the packets to be routed while stalling all others. Flow-control signals are dynamically interconnected by the crossbar in the reverse direction to the data paths, thus ensuring that flow control is preserved through the chip. When a packet is stalled data flows until all buffering along its path is filled, at which point the flow-control protocol of the DS-Links communicates the stall to the preceeding device. As soon as one packet has been transmitted from a link, the arbiter for that link allows the next one to proceed, ensuring that every packet is sent after a bounded delay. The arbitration is performed round-robbin, since although FIFO arbitration is better theoretically, in simulations the difference in performance is only a few percent, which in practice would be more than offset by the slower operation of a more complex arbiter.

A single IMS C104 connects up to 32 devices and arbitrary numbers of IMS C104s can be connected together to construct packet-switching interconnect for any number of devices. Because all of the links and packet processors of the interconnect operate independently, the performance of the system scales with its size, and because of the high valency of the IMS C104 many devices can be connected using relatively few routers, making the interconnect highly cost-effective.

We have considered the necessity for the interconnect to be scalable and flexible to maximize its range of application. To achieve this the IMS C104 is highly programmable, for instance, the routing can be varied to allow a variety of network topologies to be used efficiently. The IMS C104 performs the routing decision for each packet by comparing the value of its header with a set of address intervals and selecting the link corresponding to the interval into which the header falls. This is extremely efficient, but places some restrictions on how the terminal links of an interconnect can be numbered. However, any link output can be programmed to discard the routing header as a packet leaves the interconnect, so this restriction can be transparent to the devices using the interconnect. Any packet identification required by the receiving device can be included after the routing header. This header deletion mechanism also allows an interconnect to be constructed hierarchically, with the numbering of devices also being hierarchical, in a similar way to the local/national/international structure of telephone numbers. The fact that header deletion changes the length of the packet causes no problems because the operation of the device is independent of the packet length.

The IMS C104 supports locally adaptive routing by allowing a free choice between a programmed 'hunt group' of output links. Provided every link is connected to the same device or to an entirely equivalent device, the choice of which link of the group to use can be made on the basis of which link is available first. This maximizes performance by ensuring that there are not several packets waiting to use one link when another equivalent link is idle. The IMS C104 also supports a method for preventing interconnect hot-spots called Universal Routing. A system of IMS C104s can be programmed

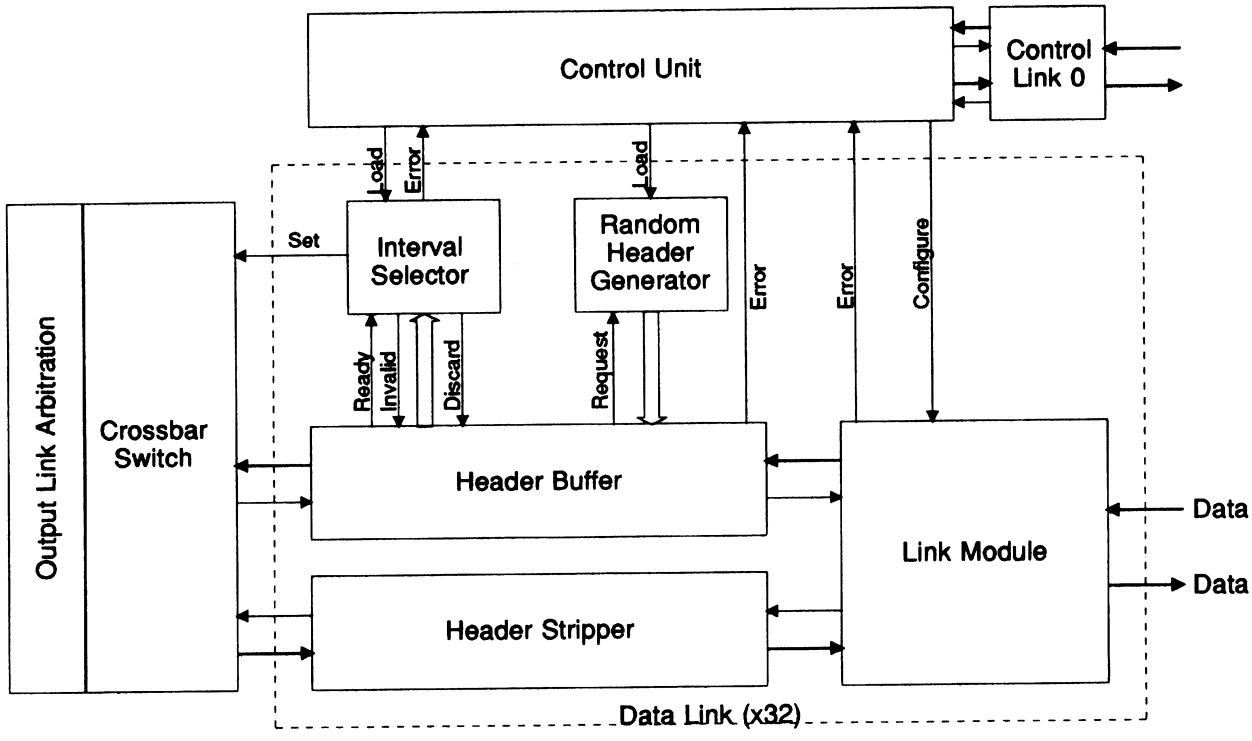

FIGURE 1. Main functional blocks of the IMS C104.

so that each packet entering it is routed first to a randomly chosen IMS C104 and from there to its original destination. By spreading the load across the interconnect this maximizes the number of links which can transport it, thereby increasing bandwidth and reducing latency under high load conditions, at the expense of peak bandwidth and minimum latency under low load.

Preliminary details of the IMS C104 are given in INMOS (1991).

5. USING CONCURRENT INTERCONNECT FOR INTERPROCESSOR COMMUNICATION

The most common component in highly parallel systems is a processor, so in this section we consider how processors can communicate most effectively using an interconnect of the type described in this paper. The construction of packets for transmission and the interpretation of packets received are both quite computationally intensive activities, particularly when the data rate of the interconnect is high and the packet size is small. Since high data rates are clearly desirable and small packets are necessary to keep latency bounded, this is a problem which cannot be avoided without compromising performance. In Ramachandran *et al.* (1990) a convincing case is made for hardware support of interprocessor communication, but there is little data available for evaluating cost. In this section we show that this cost need not be very high.

One example of a processor with such hardware support is the IMS T9000 transputer (INMOS, 1993), the general layout of which is illustrated in Figure 2. The external communications of the T9000 are managed

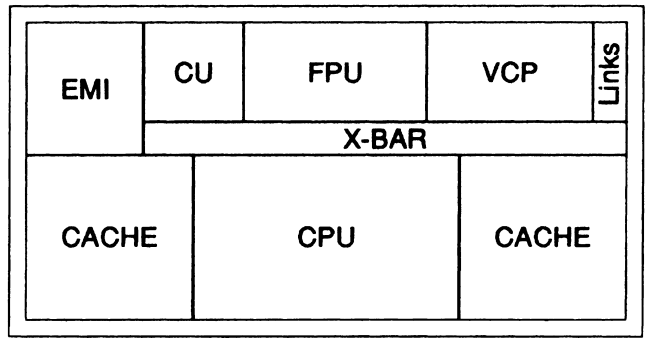

FIGURE 2. Layout of the T9000.

by a sophisticated on-chip communications processor called the 'Virtual Channel Processor' (VCP). The main function of the VCP is to accept high-level communication commands from the processor and translate them into sequences of packet exchanges on serial links obeying a strict protocol. The VCP multiplexes simultaneous communications on an arbitrary number of unidirectional channels, which couples well to a concurrent interconnect. It does this by keeping information relating to each virtual channel in a data structure in memory. Each time a packet relating to a particular channel is to be sent, the VCP adds the corresponding data structure to a linked list. As each one is taken from the head of a list, the VCP sets up a DMA transfer directly from the workspace of the communicating process using the channel to one of four serial DS-Links. Every packet starts with a header taken from the data structure and contains at most 32 bytes of data, giving a effective bi-directional data rate of upto 17.6 Mbytes/s per link.

The VCP of the IMS T9000 provides a high data rate interface to a concurrent interconnect. It interacts with the CPU to implement an extremely simple programming model of synchronized communication along an arbitrary number of logical channels, which can be connected through the interconnect to an arbitrary number of other processors and other devices. Since the VCP occupies only 16% of the total chip area, this shows that the capability of a concurrent interconnect can be exploited without excessive cost, even compared with standard busses, which generally require buffers, drivers and control logic external to the processor chip. Integrated interfaces to busses such as PCI require similar amounts of silicon as the IMS T9000 VCP (DEC, 1993).

6. CONCLUSIONS

For high-performance parallel systems to become commonplace, they require an interconnect whose cost/performance matches that of the systems themselves. Converging industry trends suggest the use of a *concurrent* interconnect, which derives its performance from the simultaneous operation of many independent communication channels. This means that the performance of each channel need not be excessively high, permitting an implementation with low-cost CMOS technology, using simple protocols on serial channels. This allows an interface to the interconnect to be integrated with another device such as a processor and allows the interconnect itself to be constructed from standard CMOS chips.

ACKNOWLEDGEMENTS

I would like to thank my present and former colleagues at INMOS for their assistance and support, particularly David May, Paul Walker and Colin Whitby-Strevens. Some of the work described in this paper has been supported by the EC ESPRIT programme.

REFERENCES

Agarwal, A. (1991) Limits on interconnection network performance. *IEEE Trans. Parallel Distributed Systems*, **2**, 398–412.

Black, U. D. (1989) *Data Networks: Concepts, Theory, and Practice*. Prentice-Hall, New York.

Clos, C. (1953) A study of non-blocking switching networks. *Bell Systems Tech. J.*, **32**.

Dally, W. J. (1990) Performance analysis of *k*-ary *n*-cube interconnection networks. *IEEE Trans. Comp.*, **39**, 775–785.

Dally, W. J. (1991) Express cubes: improving the performance of *k*-ary *n*-cube interconnection networks. *IEEE Trans. Comp.*, **40**, 1016–1023.

Dettmer, R. (1992) Frame relay: the networker express. *IEEE Rev.*, **November/December**, 381–385.

DEC (1993) DECchip 21066—alpha AXP processor for low-cost applications. In *Proc. HOTChips V*. IEEE CS, Stanford.

Gustavson, D. B. (199?) Bus guidelines and trends. In Di Giacomo, J. (ed.), *Digital Bus Handbook*. McGraw-Hill, New York.

INMOS Ltd (1991) *The T9000 Transputer Products Overview Manual*. INMOS Ltd, Bristol.

INMOS Ltd (1993) *The T9000 Transputer Hardware Reference Manual*. INMOS Ltd, Bristol.

van Leeuwen, J. and Tan, R. B. (1987) Interval routing. *Comp. J.*, **30**, 298–307.

May, M. D., Thompson, P. W. and Welch, P. H. (eds) (1993) *Networks, Routers and Transputers: Function, Performance and Applications*. IOS Press, Amsterdam.

McAuley, A. J. (1992) Four state asynchronous architectures. *IEEE Trans. Comp.*, **41**, 129–142.

Ni, L. M. and McKinley, P. K. (1993) A survey of wormhole routing techniques in direct networks. *IEEE Comp.*, **26**, 62–76.

Ramachandran, U., Solomon, M. and Verson, M. K. (1990) Hardware support for interprocess communication. *IEEE Trans. Parallel Distributed Systems*, **1**, 318–329.

Teener, M. (1992) A bus on a diet .... In *Proc. CompCon '92*. IEEE Computer Society.

Thompson, P. W. (1993) Globally connected fault-tolerant systems. In Kerridge, J. (ed.), *Transputer and Occam Research: New Directions*. IOS Press, Amsterdam.

Walker, C. P. H. (1992) The bus-less computing environment. In *Proc. BUSCON/92-West*, Vol. II, pp. 549–557. Conference Management Co., Long Beach.

Whitby-Strevens, C. (1993) *Standard for Heterogeneous Interconnect (Draft)*. INMOS Ltd, Bristol.