

# Hardware Implications of Virtual Channels, particularly for the Bus Interface

C P H (Paul) Walker

*4Links for technical help, +44 1908 566253, paul@walker.demon.co.uk*

**Abstract.** This paper suggests some special requirements for the interface between virtual channel communications and the system bus of a computer such as a PC. It examines a number of existing interface boards and chips to determine the features which best satisfy the requirements. These desired features are assembled and found to be those which minimise waste, whether of time, transfers, performance, or cost. An observation is that the optimum size of a DMA transfer burst is about the same as the payload of an ATM cell or of an IEEE P1355 DS-Link packet. An outline interface design is proposed and reviewed which appears to incorporate the best of the existing interfaces in meeting the requirements for multiple virtual channels. Implications of the interface are examined on the network topology, which suggests advantages for an interface with two external ports.

## 1. Introduction

There is now a whole family of communications interfaces designed to handle communications to different destinations or to different processes at a destination concurrently down the same physical connection. Each concurrent communication appears to have a dedicated channel, and so is described as a 'virtual channel'; in practice the virtual channels are multiplexed by hardware or software onto the physical connection.. The most obvious recent example of virtual channel communications is ATM (Asynchronous Transfer Mode or perhaps originally Asynchronous Time-division Multiplexing) [1, 2]), developed by the telecomms industry and being aggressively developed for computers and datacomms. Fibre Channel [3] and SSA (Serial Storage Architecture) [4] have been developed principally for disk drives and systems, but have wider application. INMOS/SGS-THOMSON's DS-Links [5], originally developed for inter-processor communication for the T9000 [6], are now being standardised as part of IEEE P1355 [7]. DS-Links also have much wider application — perhaps because all equipments now contain processors and any communication is between these processors.

The transputer user community ought to have a head start in the use of virtual channels, because they have been using multiplexing processes for the links of transputer arrays for many years. In the last few years these hand-crafted processes have been replaced by virtual channel configuration software from INMOS and from third parties. The T9000 family including the C104 and C101, with its built-in hardware for virtual channel communication, perhaps provides more complete support for virtual channels than any of the other virtual channel developments.

Unfortunately, the world does not consist only of transputers, let alone T9000s, and so a means must be provided to interface virtual channels to conventional hardware such as

PCs and workstations. The existing transputer interface boards are far from ideal for this purpose. Most existing PC adaptors for other interfaces are equally unsuitable, because they are mostly designed for a single channel of communication.

In order to find an interface closer to what is needed, this paper offers a condensed version of a product-definition process: it lists requirements, examines a number of existing products to see their strengths and weaknesses with respect to the requirements, extracts the features closest to the requirements, synthesises these features into an outline of a proposed design, and briefly reviews the proposal against the requirements.

## 2. Requirements of a Virtual Channel hardware interface

The following table includes needs and wants the author has collected from transputer users, from ATM developers, from published articles, from postings to Usenet, and from transputer designers. No priority or weightings are implied, and no attempt has been made to remove inconsistencies or conflicts.

Table 1: Needs and Wants for a Virtual Channel Hardware Interface

<i>Cost</i>	<i>Function</i>
Don't want to pay more than twice Ethernet (10BaseT) board cost	Want 'Multimedia'
Want board for PC to cost less than \$100	Want ten to twenty channels on a terminal
Don't want to pay for memory that can't be used	Want 100 to 1000 channels on a server
	Want arbitrary number of channels
	Want separate I/O to processor, to disk, and to video
<i>Performance</i>	<i>Pain</i>
Want sustained bandwidth better than any Ethernet	Want it to be transparent to existing comms software when used for single channel
Want enough bandwidth for TV quality compressed video (about 7 Mbits/s)	Don't want to dismantle the PC to install the board
Want peak bandwidth at least ten times Ethernet	Don't want to buy a new PC to use the interface
Want 100% utilisation of 155 Mbits/s full-duplex	Want to plug and play without reading documentation
Want fast response	Want same board used in both server and terminals
Want latency to be hidden by 'excess parallelism'	Want it available yesterday
Don't want I/O to slow down the processing	Want board to be able to cope with evolving standards
Don't want heavy processing to slow down the I/O	
Don't want to shuffle data around memory	

## 3. Some example interfaces to existing transputer (Txxx)

### 3.1. INMOS B008 and clones

The IMS B008 [8] uses a single C012 link adaptor [9]. The C012 is accessed as a slave on the ISA bus, where it is mapped onto four locations in the PC's I/O space, with another four locations used for system control. As well as the slave access of the C012, the B008 can generate interrupts for the PC, and it can also use the PC's DMA mechanism.

When the B008 was designed, DOS was very firmly a single-process operating system, and so the only thing a program on the host PC could do was to drive the B008. Using the interrupts or DMA only increased the PC's idle time, with no noticeable performance improvement. Apart from this, the C012 had two channels, one in each direction, and it proved difficult to set up DMA transfers for both directions. As a result of the lack of perceptible performance increase, and the difficulty of bi-directional DMA, the B008's DMA mechanism has rarely been used. With just the slave interface, throughput is limited to about 500 kBytes/s, less than can be achieved with Ethernet.

A number of third-party manufacturers of transputer boards have attempted to improve on the B008 interface, usually by adding data buffering (as FIFO, as shared memory, or as the internal memory of a transputer) and by using the full 16-bit bus width.

### *3.2. FIFO*

A FIFO is useful in allowing both the bus and the transputer to operate at their full speed, without having to synchronise on each byte transferred. The FIFO interface to the bus can be made 16-bits wide, thus doubling the throughput on the bus.

### *3.3. Shared Memory*

More flexible data buffering can be provided by shared memory, which perhaps can be large enough to buffer a complete message.

In principle, data does not need to be moved between the shared memory and the processor's main memory. In practice the data almost always is moved. The move has to be done by program, which slows down the user's application.

A buffer large enough to hold a complete message means that the processor only needs to be involved once per message, but it also means that each message has to be completely transferred into the buffer before it is transferred out of the buffer. For small cells, such as 32 or 48 bytes, this is no problem, but for a long message it adds greatly to the message delay.

The shared memory adds cost far exceeding the cost of the equivalent amount of main memory, and is not normally usable for any purpose except link transfers.

So, while initially appearing attractive, shared memory interfaces reduce performance and increase cost, compared with an optimum interface.

### *3.4. Handshaken bus access*

The B008 uses a 16-bit T225 transputer to control the C004 link crossbar switch. One company has used this transputer rather more effectively by giving this transputer access to the ISA bus [10]. The block diagram looks very much like a shared memory design, but there is no shared memory. To send data to the transputer, the PC does a write access to the bus and the transputer does a read access to the same bus; each access waits for the other, and when both accesses are present the transfer is completed. For data from the transputer to the PC, the PC does a read access and the transputer a write access, which synchronise in the same way.

The peak transfer rate claimed for this interface is 3.5 Mbytes/s, a substantial improvement over the B008 and its clones, for negligible additional cost. Against this, the PC's processor is fully occupied during the transfer, and any synchronisation failure might cause problems.

### 3.5. Transputer as bus master

Although the INMOS B008 has DMA logic, the DMA capability has rarely been used. A major reason for this lack of use is that there is only one DMA channel available at once, whereas the B008 needs at least two channels — one in each direction. With the many virtual channels required in this study, the PC's single channel per I/O board is an even more serious limitation.

Fortunately, the ISA bus offers a means of bypassing the PC's DMA chips except for arbitration. After the board is granted the bus, the board asserts a 'Master' signal and then provides memory address and data along with the appropriate timing signals. In providing the address, a different set of addresses can be used for each of many channels — for example a different channel could be used for each direction of each link of a transputer.

Peak transfer rate for a burst DMA in Master mode is a 16-bit word every 350 ns, or 5.7 MBytes/s. How much of this peak rate can be utilised depends on the burst length and the amount of hardware assistance for the transfer.

The master mode DMA appears to offer the highest bandwidth, with any number of channels, with much less interference on processor performance than the other interfaces, and without needing shared memory. Although it is widely used for Ethernet and for SCSI, the author knows of only one implementation with transputers or transputer links [11].

## 4. Interfaces other than transputers

### 4.1. AMD PCnet-ISA Ether net interface

Much can be learnt from interfaces such as for Ethernet which may have been designed without reference to virtual channels or to the short cells of ATM and IEEE P1355 DS-Links.

Like the bus master transputer interface just described, the Am79C960 Ethernet chip [12] provides remarkable performance from the bus, without requiring a large buffer memory. Instead of relying on the PC's DMA controller to provide the address and length count for the transfer, this chip just uses the DMA controller to give it access to the bus, and then proceeds as bus master. When granted the bus, it transfers sixteen words of 16 bits at about one word every 350 ns. Allowing for the latency of arbitration for the bus, and for a small number of overhead accesses, this is equivalent to about 5 Mbytes/s. The choice of sixteen words, 32 bytes, implies a transfer lasting about 7  $\mu$ s, which is a compromise between:

- \* transferring enough bytes that the overheads of arbitration and of other accesses do not increase latency to the point where buffers overflow
- \* keeping the FIFO buffers in the chip as small as possible (136 bytes for transmit, 128 bytes for receive)
- \* completing the transfer soon enough (in less than 15  $\mu$ s) that a refresh access can occur, and that the processor is not delayed for too long.

The 10 Mbits/s of Ethernet means that at full Ethernet utilisation, the chip needs only 1.25 Mbytes/s from the bus — a utilisation of 25% — which allows the processor and bus to do something other than transfer to Ethernet. (Full-duplex use of ATM or IEEE P1355 DS-Links at 25 Mbits/s would need a bus bandwidth — for the *payload* — of about 5 MBytes/s, depending on which ATM Adaption Layer (AAL) or on the length of

P1355 packets. As the ISA bus is limited to 5 MBytes/s, there would appear to be little point in having an ISA bus interface connected at a higher speed than 25 Mbits/s full-duplex.)

#### *4.2. A Virtual Channel FIFO*

In an attempt to offer an improvement to the bus interface, the author proposed a 'Virtual Channel FIFO' [13], and test marketed the proposal with a number of transputer users. It provided 16 channels in each direction, with 32 Kbyte FIFOs for each channel, to allow a total buffer of about 50 millisecond's worth of link traffic at 100 Mbits/s.

The test marketing showed that some form of FIFO buffering is needed, but that it is much preferable for the large buffers to be in main memory rather than in special memory on an I/O board. The reason for wanting large buffers is that the operating system can not cope with frequent system calls, and so a number of messages should be buffered to be handled by a single system call. The many milliseconds latency of the OS, however, must not be confused with the latency for a Bus Master DMA access, which is only a few microseconds.

In fact, as with any form of shared memory, the net effect of a large Virtual Channel FIFO is to add cost and reduce performance.

#### *4.3. The TranSwitch SARA evaluation board*

TranSwitch make a range of telecomms components and have recently launched ATM chips including a chip-set [14] for Segmentation And Re-Assembly (SARA) of packets to and from cells. (Here 'packet' is used in the LAN sense of several kBytes, more equivalent to a transputer link 'message', and 'cell' is used in the ATM sense of 53 byte cells, more equivalent to a transputer link 'packet'.)

As well as its basic purpose of splitting packets into cells for transmission, and combining cells into packets on receipt, the SARA chips perform a substantial and useful part of any ATM node interface, such as adding and stripping headers, computing and checking CRCs, and scheduling the cells. The chips can handle a queue of up to 8000 messages, and can handle anything from 512 to 64K virtual channels.

To help customers gain experience with the SARA chips, TranSwitch have produced an evaluation board (for the EISA bus). The board uses four different sets of RAMs, two for each direction: SRAM for the control information such as message descriptors and virtual channel parameters; VRAM for storing the data packets.

For a first product as an evaluation board, the use of VRAM for storing the data packets is appropriate. But if the lessons of the bus master interfaces and of the virtual channel FIFO apply, it may be expected that future versions of boards with SARA chips dispense with the VRAM, to give higher performance, lower cost, and more efficient use of memory.

#### *4.4. The TI/SUN ATM chip-set*

Constraints are sometimes an advantage, and the small board size of SBus [15] boards means there is not much space for a large buffer memory. The TDC1500/1560 is a two-chip set [16] developed jointly by TI and SUN for interfacing ATM to SBus, and the only other components needed are a small SRAM (to store control blocks) and small EPROM (to store initialisation code).

The chip-set provides full duplex ATM at 25 to 155 Mbits/s, with 1023 virtual channels (and corresponding DMA channels) for receive and 255 DMA channels for

transmit. It does master-mode DMA bursts of up to 32 bytes per burst, and includes a receive FIFO for 32 cells and a transmit FIFO for 8 cells.

Full support is provided for AAL5 (the ATM Adaption Layer which uses the full 48 bytes of the cell payload as data), but handling of AAL3 and AAL4 (the adaption layers which include local sequence numbers and checksum within each 48 byte payload) is left to the processor. Transmit data can be taken from wherever it is in memory, with no requirement for byte alignment, but receive data is aligned to a 16-byte boundary. Compared with some of the problems with other interfaces these are minor deficiencies, and the chip-set would appear to offer both high performance and low cost.

#### 4.5. *VLbus and PCI*

Similar bus performance to SBus is provided for PCs by VESA Local Bus and PCI (Peripheral Component Interconnect bus) [17], which both overcome the limitations of ISA and particularly its peak transfer rate of 5.7 MBytes/s. PCI runs at 33 MHz and can transfer 32 or 64 bits every clock period — a peak data rate of 133 or 267 MBytes/s. Of course not every cycle can actually be used for data, and slow devices may use an excessive proportion of the available bandwidth. But the data transfers associated with payloads of ATM or DS-Links, even at 200 Mbits/s full duplex, are only 15% to 30% of the bus bandwidth.

In spite of PCI's high bandwidth, and of its normal latency on a lightly loaded bus of less than a microsecond, the spec. advises using a value of 30 ns for the guaranteed access latency, and recommends 50 bytes of buffering for the 10 Mbits/s of Ethernet and 500 bytes of buffering for the 100 Mbits/s of FDDI. This amount of buffering is still far less than is needed to buffer the delay of the operating system.

#### 4.6. *VMC*

With the growth in multi-media, a need has appeared for access to a graphics frame store by other devices rather than just the display controller chip. Because access across the system bus (even VL or PCI, let alone ISA) is too slow, designers have been forced to duplicate the frame store.

In the examples that have been examined of access to the bus, we have found that duplicating the memory increases cost and reduces performance compared with the optimum approach. Indeed the same appears to apply to the use of the frame store, if a dedicated bus can be used for the frame store accesses.

A bus which allows the use of a single frame store is the VESA Media Channel, VMC [18]. By providing fast access to the frame store, it minimises the buffering required on boards that access the frame store, as well as increasing the performance of these accesses. By keeping this traffic away from the main system bus, the bus and processing performance are also improved.

Of course for a communications interface such as ATM or DS-Links, both of which are likely to be heavily used for multi-media, the VMC bus is an extra interface. But it could well prove to be worth-while to include both VMC and system bus interfaces on the same board, or to have separate link nodes for the two interfaces.

## 5. The T9000 and C101

### 5.1. The T9000

For DS-Links, a small proportion (16% [19]) of the T9000 chip [6] performs the function that the two TranSwitch SARA chips [14] plus considerable extra logic perform for ATM. Indeed it may be useful to view the T9000 as a communications device with four virtual channel communications links capable of an aggregate throughput of 64 or 128 Mbytes/s — and take the integrated processor as a bonus.

It would be possible to use the T9000 for Bus Master access, just as the example above with a 16-bit transputer. As with the example, however, it would be useful to put a small amount of FIFO buffering between the T9000 and the bus, so that the transputer and bus do not have to synchronise on every cycle. Such external buffering does not prevent the T9000's cache being used as additional data buffering, this time decoupling the link transfer from the external memory interface. In particular, the cache (or on-chip RAM) is invaluable for storing the Virtual Link Control Blocks (VLCBs).

This is because for every 32-byte packet the T9000 transmits or receives, it accesses eight words (32 bytes) of VLCB. As the VLCB includes a linked list of packets to send, these accesses are not just to a single VLCB but must be to at least two VLCBs, so there is actually more work in accessing the control information than there is in accessing the data.

For the T9000, with its on-chip cache capable of supporting up to 400 Mbytes/s (25 MHz CPU), the cost of these control accesses is negligible. If the accesses had to go via a bandwidth-limited system bus (VLbus, PCI and SBus as much as ISA), such an overhead would not be acceptable. The control information must therefore be held as locally as possible to the link interface.

### 5.2. The C101

A lower cost solution than a T9000 for one or two P1355 DS-Links is offered by the C101 [20] which connects a link to a microprocessor bus, and the user can choose whether the bus is 8 bits, 16 bits, or 32 bits. Choice can also be made as to whether the data is available on a separate FIFO like port, or on the bus. There are internal FIFOs for each direction, enough for two packets for the T9000 or just over one cell for ATM.

Together with providing separate ports for data and control, the C101 optionally separates the functions of processing the headers and transferring the data. This allows concurrency between header processing and data transfer. While the header is being processed (to decide where the associated data should be transferred) the data is filling the FIFO. The data transfer can then be set up, to take place while the next header is being processed.

An additional contribution of the C101 to ease system design is its use of a clocked interface, which is in tune with the clocked interfaces of faster FIFOs and newer memory chips, and with the newer buses such as PCI and VMC.

## 6. Aspects of optimising system performance

### 6.1. Generic lessons

The lessons learnt from these example interfaces are in many respects the normal lessons of engineering and quality, that one should eliminate waste.

For example, eliminate:

- \* Unnecessary accesses across a bottlenecked system bus
- \* Unnecessary duplication of memory, leading to unnecessary transfers between the two memories as well as the unnecessary cost of the redundant memory
- \* Unnecessary serialisation where one part of the system is idle waiting for another part of the system, and then vice-versa
- \* Unnecessary interference by the processor on I/O performance
- \* Unnecessary interference by the I/O on processor performance

### 6.2. Specific lessons for an I/O board for virtual channel communications

The following points can be gleaned from considering how the different interfaces meet the requirements and meet the generic aspects of minimising waste:

- \* Store control blocks locally on the I/O board if they have to be accessed for every cell, to avoid unnecessary accesses over the system bus (T9000, TranSwitch, TDC1500/1560)
- \* Use main memory for any large data buffers required by the system processor (PCnet-ISA, Transputer bus master, TDC1500/1560)
- \* Use video memory for any large data buffers required for display (VMC)
- \* Minimise the amount of data buffering on the I/O board; no more should be necessary than for a single channel, although its organisation may be different from a conventional FIFO (Transputer bus master, PCnet-ISA, TDC1500/1560, *vs.* Virtual Channel FIFO)
- \* Use master mode DMA, with the I/O board providing the addresses for the transfer (Transputer bus master, PCnet-ISA, TDC1500/1560)
- \* Use FIFO like buffering to allow bus transfers to take place concurrently with links (most examples except B008)
- \* Size the FIFOs to buffer the latency of DMA, rather than the latency of the operating system (PCnet-ISA, PCI, TDC1500/1560 *vs.* Virtual Channel FIFO)
- \* Synchronise the FIFOs on packet/cell boundaries rather than on byte or word boundaries (Virtual Channel FIFO)
- \* Do the DMA transfers in complete packets/cells to avoid an unnecessary extra layer of segmentation and reassembly (TranSwitch)
- \* Allow for variable length cells in the FIFO, even for ATM, in order to provide for different and as yet undefined ATM Adaption Layers (AALs) (C101, *vs.* TDC1500/1560)
- \* Include the termination flag in with the data FIFO, so that hardware can handle termination of short cells/packets (C101)
- \* Include the memory address with the data FIFO, but provide a separate path to it so that the address can be processed concurrently with the data (PCI, C101)
- \* Separate the header processing from the data transfer, so that the header can be processed concurrently with data transfer (C101)

- \* Use a linked list of messages, and only interrupt when the list has been empty and becomes non-empty (T9000, TranSwitch, vs. TDC1500/1560)
- \* Use a clocked, synchronous interface (PCI, C101, fast FIFOs, TDC1500/1560).

## 7. An outline interface design

At least some of the lessons learnt from the examples examined are incorporated in the outline design shown in Figure 1. The four principal blocks in the diagram are:

- \* A Dual-Port RAM to hold the control blocks:  
If each control block is the same size as the 32-byte VLCBs of the T9000, a memory of 512 Kbytes is enough for 64 K channels; alternatively 32 channels can be serviced by a low-cost 1 kByte memory
- \* A microprocessor:  
If the program can be kept simple enough, it may be possible to implement the program in a hardware state machine, but even with such a state machine an inexpensive processor is useful for initialisation and error recovery
- \* The serial interface chip:  
The example shown has the same ports as the C101; other serial chips for ATM, Fibre Channel, or SSA, would have broadly similar connections
- \* A synthesised block described as a 'Cell FIFO':  
The Cell FIFO performs the function of a small bi-directional FIFO, but which includes a few modifications from a standard FIFO as a result of the lessons learnt from the examples. The Cell FIFO is described in more detail after the system block diagram.

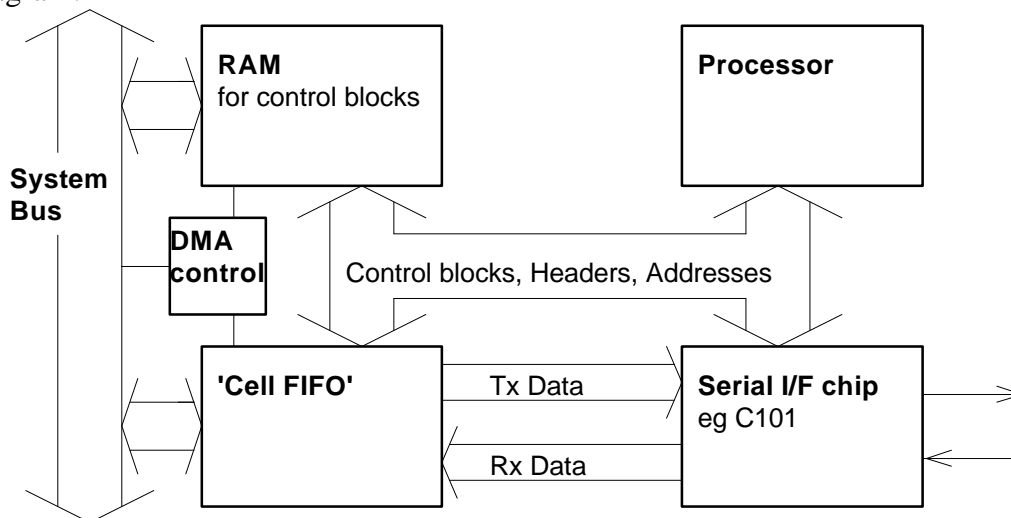


Figure 1: Block diagram of proposed bus interface for virtual channels

Each element of the cell FIFO consists of:

- \* Data storage for the maximum number of bytes in a cell (ATM) or packet (DS-Links), up to a maximum of 64 bytes per cell/packet
- \* The main memory start address to which the data of the cell/packet is to be written (Rx) or read (Tx).
- \* A termination flag associated with each data byte (if cells/packets can contain any number of bytes) or with each word (if all cells/packets are a whole number of words)

A transmit element of the cell FIFO is valid for DMA if the address has been written to the element and if the data contents of the element are empty.

A receive element of the cell FIFO is valid for DMA to the system bus if the address has been written to the element and if either all the data bytes have been written or a termination flag has been written.

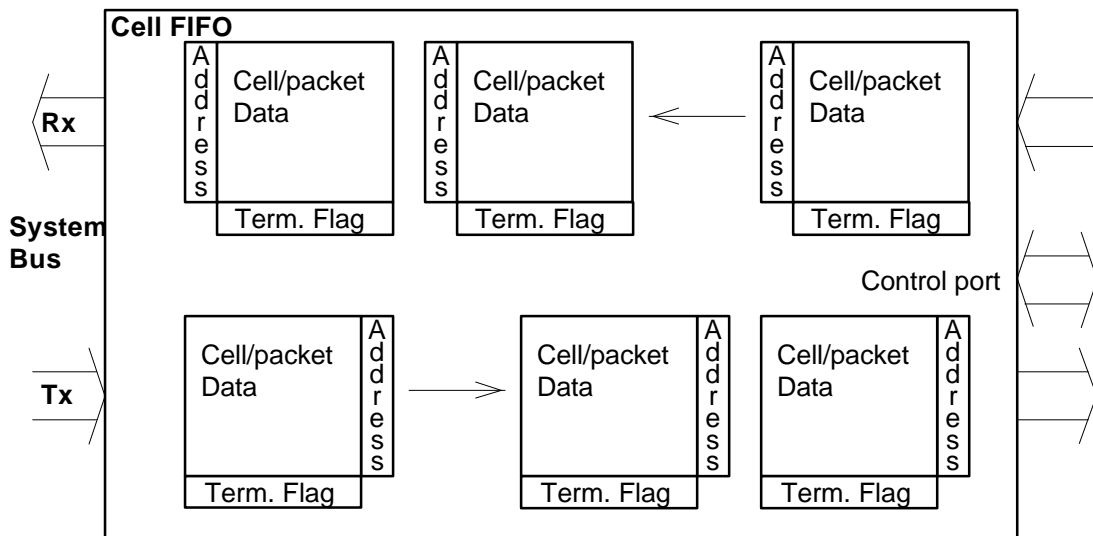


Figure 2: Block diagram of Cell FIFO

The size of the cell FIFO is chosen to overcome the latency of DMA, which may be from four cells/packets up to 16 or more in each direction, depending on the performance needed by the links and provided by the system bus. This memory of a few hundred bytes could easily be integrated within an interface chip for one of the standard buses such as ISA or PCI.

The amount of memory used in the cell FIFO could be less for DS-Links than for ATM, because buffering for ATM needs to cover the worst case DMA latency, whereas the low-level flow-control of IEEE P1355 DS-Links means that data will never be lost as a result of a longer than normal latency, so the buffering can be designed for the average latency rather than worst case.

## 8. Review of the proposed design

The proposed design meets many of the requirements, but by no means all of them. Fortunately, most of the requirements not met by the basic design can be met by choices taken in implementation.

### 8.1. Cost

The principal determinant of cost is volume such that most of the interface can be integrated onto a small number of chips. Avoiding the use of DRAM, and particularly VRAM, and using small FIFO buffers, should minimise cost, and the proposed interface does not require a great deal more functionality than is present in the AMD PCnet-ISA chips.

### 8.2. Performance

Performance up to at least five times Ethernet should be provided by any interface board, including those with the ISA interface.

Those users who need 100% utilisation at 155 Mbits/s must use one of the other buses such as PCI, with the proposed circuit implemented 32 or 64 bits wide. The proposed

interface should provide low latency, with minimal interference between processing and I/O, and with minimal data shuffling around memory.

### 8.3. *Function*

Different numbers of virtual channels can be handled with different amounts of memory for the control blocks. If there is a problem with having 'magic numbers' of channels above which a radical change is required, and if performance is permitted to degrade as the number of active channels is increased, it might be preferable to implement the control block memory as a small cache.

Minimal interference between the different interfaces could be achieved by integrating VMC and SCSI interfaces, both to the system bus and to the virtual channels, on a single adaptor board — but it is probably preferable to interface the system bus first.

### 8.4. *Pain*

It is unlikely to be possible to preserve complete compatibility with existing software when the interface is used (rather wastefully) as a single channel, but preserving the same register structure might be useful for minimising the changes required.

The only way to avoid dismantling the PC to add a new interface board is to use PCMCIA.

If a user does not want to buy a new PC to use the board, the one interface which dominates existing PCs is the ISA bus. Even though this limits the performance, it still offers performance from virtual channel interfaces much higher than from existing Ethernet.

Plug and Play is provided by PCI and PCMCIA, which may be a reason for implementing the proposed interface to these buses.

The proposed interface uses minimal RAM, allowing users access to memory used by virtual channel I/O when there is no such I/O taking place.

Servers tend to need substantially higher bandwidth than terminals, and the ideal adaptor board for a server would include a small routing switch acting as a hub. An adaptor board for PCI might be appropriate with such a routing switch for the hub/server, while an ISA adaptor board is used for the terminals.

Need for a product available yesterday suggests that an appropriate methodology is to compile programs into hardware such as with the Oxford University tools for developing special-purpose processors or co-processors in programmable logic [21].

Need to cope with evolving standards means using programmable logic which can be on-board reprogrammed, such as SRAM-based or Flash-based, and using Flash memory for any ROM used by the control processor.

In the short term, the use of re-programmable logic may imply higher costs than are desired, but the opportunity to build and rigorously test the design in this form provides an excellent basis for subsequent integration within a small number of dedicated chips.

## **9. Implications of the bus interface on the network**

### *9.1. A hierarchical network with ISA boards connected by hubs*

The ISA bus is the most pervasive existing bus and is effectively limited to a serial link speed of about 25 Mbits/s full-duplex (or perhaps 50 Mbits/s half duplex). As this performance from a dedicated link gives a terminal probably at least an order of magnitude improvement over Ethernet, and is adequate at least for compressed video, it ought to be adequate for many users. Combining this with the constant bandwidth-distance product of serial links means that the cable to the terminal should also run at this speed — if it runs faster it will just carry more idle cells and will not be able to carry them as far or as reliably. This, in turn, suggests a need for small multiplexing hubs, which take a number of 25 Mbit/s links and multiplex them to a single 100 or 200 Mbit/s link.

Indeed there may be a need for a hierarchy of hubs and switches, with the cables between them operating at the lowest possible speed and with higher speeds available where necessary to minimise the number of cables between switches.

### *9.2. A direct network without hubs, and with bandwidth shared (or wasted?)*

An alternative network topology is a 'direct' network where each terminal connects to two or more other terminals, and each terminal provides routing for cells destined for another terminal. A disadvantage of these direct networks is that each terminal has to provide bandwidth not only for its own data but also for (possibly many) other terminals' data. If there is sufficient excess bandwidth available, however, and the terminals are sufficiently close together that the cost of that bandwidth is not excessive, this may be an appropriate topology. In this case, an interface board to ISA could well benefit from the use of 100 Mbit/s links in order to provide the bandwidth for the other terminals.

### *9.3. Hubs and short rings*

An attractive compromise between the direct and indirect (hubbed) networks is to provide two link interfaces on each adaptor board. These can be taken to different hubs both to give higher bandwidth and fault tolerance; or they can be connected in a ring with the bandwidth shared across the ring; or they can be connected in short rings where each end of the ring terminates at a different hub — which preserves the fault-tolerance, keeps cables short, and shares bandwidth where more is available than is currently needed by the terminals.

## **10. Conclusions**

We have learnt from the existing interfaces and attempted to synthesise a design which comes much closer to meeting the requirements. In particular, the interface should store control information for each channel, but should minimise the amount of data buffering; bus transfers should be by master-mode DMA, with the transfers the same size as the cell payload. The design should be done quickly, preferably with automated programmable logic tools, before integrating the design in silicon. A two-port board has a number of advantages.

## 11. Acknowledgements and references

I am grateful to the people and companies who have provided information on the products and interfaces used as examples, and particularly to Peter Thompson of INMOS, David Boreham of Network Designers Limited, Flemming Christensen of Sundance Multiprocessor Technology, and Michael Poole, parallel programming consultant, for their encouragement and feedback on previous proposals.

Comments on this paper will be most welcome.

Note: Several of these references are to data sheets or to drafts of standards documents. The version quoted is the latest version the author has seen. Later versions may be available from the referenced source.

- [1] Martin De Prycker, *Asynchronous Transfer Mode, solution for broadband ISDN*, Ellis Horwood, 1991.
- [2] A G Fraser, *Early Experiments with Asynchronous Time Division Networks*, IEEE Network, Jan. 1993.
- [3] ANSI X3T9.3, *Fibre Channel: Physical and Signalling Interface (FC-PH)*, June 1992.
- [4] IBM Corporation, *Serial Storage Architecture*, Dec. 1991.
- [5] May, Thompson & Welch (eds.), *Networks, Routers and Transputers*, IOS Press, 1993, ISBN 90 5199 129 0.
- [6] INMOS/SGS-THOMSON *T9000 Hardware Reference Manual*, DBT000RMST/1.
- [7] IEEE, *Draft Std P1355 Standard for Heterogeneous Interconnect (HIC)* rev. D.01, available from Bull or INMOS.
- [8] INMOS, *IMS B008 User manual*.
- [9] INMOS, *IMS C012*, in *The Transputer Databook, 3<sup>rd</sup> Ed.*, 1992, SGS-THOMSON, DBTRANST/3.
- [10] Transtech Parallel Systems, *TMB16 Hardware manual*.
- [11] Sundance Multiprocessor Technology, *SMT012 Product Overview*.
- [12] AMD, *Am79C960 PCnet-ISA™, Technical Manual*, BAN-12M-3/93-1, May 1992.
- [13] Paul Walker Consultancy, *Virtual Channel FIFO*, Sept. 1993.
- [14] TranSwitch, *SARA Chipset Technical Manual*, TXC-95000, Ed. 3, May 1993.
- [15] Sun Microsystems, *The SBus Specification*, Revision A, Sept. 1989.
- [16] Texas Instruments, *ATM Interface Products: Product Information*, SCYDE 03, Jan. 1994.
- [17] Intel Corporation (PCI Special Interest Group) *PCI Specification* Rev. 1.0, 1992.
- [18] Video Electronics Standards Association, *VMChannel 1.0P Proposal*, Sept. 1993.
- [19] Peter Thompson, *Concurrent Interconnect for Parallel Systems*, The Computer Journal, V36, #8, 1993.
- [20] SGS-THOMSON, *IMS C101 parallel DS-Link adaptor*, Preliminary Data, Sept. 1993, 42 1593 01.
- [21] Ian Page, *Automatic Design and Implementation of Microprocessors*, in *WoTUG-17 Progress in Transputer and occam Research*, eds. R Miles and A Chalmers, IOS Press, 1994.