Technical Report

UCAM-CL-TR-219
ISSN 1476-2986

Number 219

Fairisle project working documents

Snapshot 1

Ian M. Leslie, Derek M. McAuley,
Mark Hayter, Richard Black, Reto Beller,
Peter Newman, Matthew Doar

March 1991

ISSN 1476-2986

FAIRISLE PROJECT WORKING DOCUMENTS
Snapshot 1
March 27, 1991

This report contains the current versions of the documents associated with the Fairisle project. These include both papers and draft documents. This collection of documents was made on March 27, 1991. Updated versions will be issued with later snapshot numbers, these will replace earlier versions.

Contents

Fairisle: A General Topology ATM LAN

DRAFT

Ian Leslie and Derek McAuley
University of Cambridge Computer Laboratory

An experimental general topology local area network based on Asynchronous Transfer Mode (ATM) is described. This network is intended to be used to support multiservice traffic. Thus provision of guarantees of quality of service to various traffic types is an important feature of the network. In order to carry on quality of service experiments, one must have both a network which is inherently manageable, and the flexiblity to provide different management strategies.

Fairisle is supported by the SERC under Grant GR/F 6090.8 and by HP Laboratories Bristol.

1 Introduction

Fairisle is a research project investigating the architecture and management algorithms for a general topology ATM network which is to be used as a private or local area network.

Fairisle was begun in October 1989. It arose from work in the Computer Laboratory in ATM networks, multimedia communications, protocol architectures, host interface design and fast packet switching. The overall purpose of Fairisle is to investigate *networks* as well as switches. Such an investigation is as much concerned with how components fit together as it is with network components themselves.

In the tradition of the Computer Laboratory this investigation is based on experience of a real network moving real user data. Thus an early problem that Fairisle has faced is in designing and constructing components. These components will form the basis, or rather the experimental apparatus, on which the real experiments will take place.

A major goal of the Fairisle network is to support multiservice traffic, supplying guarantees when required (e.g. on bandwidth, delay, cell loss). Decisions about what sort of guarantees are made and how many types of guarantee there are, have been left open for experimentation. It is the management of network resources which will be the most novel aspect of Fairisle.

In developing an network architecture which is to be used as a basis for experimentation in network resource management, one must produce an architecture that gives rise to a *manageable* network. By a manageable network we mean one in which both network resources and resource demands made by traffic are identifiable and quantifiable. The precision of guarantees given to different types of traffic will necessarily be related to the precision with which traffic demand is specified when communication is initiated.

As well as an architecture, we have developed an implementation of the network. This implementation will eventually include specific management algorithms, but in the first instance it must be sufficiently flexible to allow a variety of algorithms to be used and monitored.

The Fairisle Network Architecture is presented in the next section. Following this, section three describes the network implementation in terms of the network components. Section four outlines how devices will be attached to the network. Finally the current status of the implementation and some concluding remarks are made in section five.

2 Network Architecture

The most obvious features of the network architecture used on Fairisle is that is it ATM based and that it uses virtual circuits.

ATM Motivation

Fairisle is an Asynchronous Transfer Mode[1] network. We are using ATM for a number of reasons. Briefly these are:

[1]We consider ATM to be a generic term, that is, meaning the use of fixed sized cells in order to simplify switching, and make the performance of switches and the multiplexing more predictable.

1. ability to carry a variety of traffic
2. that a B-ISDN based on ATM will one day be the preferred wide area interconnect
3. our experience with ATM
4. that future network traffic demands are, and will continue to be, unpredicatble

The arguments for using ATM to provide network integration are well known (as are many of the counter arguments). However, we also believe that providing a continuim of services from delay insensitive traffic (such as traditional data) through to constant rate delay sensitive traffic (such as traditional voice) will be important for future networks. A time critical remote procedure call is an example of a communication which lies somewhere in this continuim rather than at one of the endpoints.

The expectation that B-ISDN will be the ubquitous high speed long haul interconnect, leads us not only to believe that the local distribution of B-ISDN traffic is important, but also that ATM provides an opportunity to integrate not just traffic types, but local and wide area communication. This has implications for network architecture — we are interested in ATM internetworking.

Finally, whatever the problems to be solved, and the penalties to be paid in running an ATM network we believe that the unpredictability of traffic demands, both in quality and quantity make ATM an attractive strategy. ATM provides a flexible network interface.

Virtual Circuit Motivation

Fairisle is a virtual circuit (or perhaps more accurately a lightweight virtual circuit) network. Each ATM cell carries with it a virtual circuit identifier. Our reasons for choosing a virtual circuit approach are as follows:

1. VCIs in cells can be processed via very simple and fast hardware lookups in the switches
2. VCIs can be used to associate a cell with an arbitrary quality of service description (held at each switch)

3. VCIs (when allocated by the end system) can be used to simplfy the implementation of the end system interface
4. a virtual circuit is an obvious unit to which to attach a guarantee and which to police.

Broadband ISDN has come to a similar conclusion for similar reasons, although one could argue that virtual circuits are part of the ISDN culture.

We have considered approaches which do not use virtual circuits. One can imagine globally addressing each ATM cell and using fields in the cell header to indicate traffic type which could be used to facilitate the management of guarantees. However, using fields in the cell header restricts the number of different qualities of service that can be provided. Moreover, the virtual circuit approach is more natural, particularly when it comes to measure resource usage against resource guarantees.

2.1 The Multi Service Network Architecture

The protocol architecture used in Fairisle, the Multi Service Network Architecture (MSNA) is more fully described in [McAuley 89].

MSNA is designed to support multiservice communication all the way up to the multimedia application. It achieves this in two ways:

1. by providing, where desired, application to application ATM cell communication, and
2. by streamlining (that is avoiding) multiplexing in end systems.

Providing application to application ATM cell communication means that applications are not constrained to use an adaptation layer which takes a view as to how communication should be handled. For example, if segmentation and reassembly are not required they are not used. An immediate consequence of wishing to provide application to application ATM cell communication is that MSNA must provide an internetworking layer which works on the basis of cell forwarding.

There are costs associated with multiplexing — these are often listed as bandwidth, processing and complexity [Tennenhouse 89], but in a multiservice environment multiplexing also introduces contention among traffic streams. The streamlining of multiplexing minimises these points of contention (although some may always exist) in end systems. This minimisation is important as each contention point places constraints on the scheduler within the end system; for example, in a real time system these constraints may require the scheduler to work much harder to find feasible schedules, or worse, cause missed deadlines. In MSNA, the virtual circuit identifier (VCI) can be used to identify the end application entity or indeed a single thread within the application entity. Thus it is possible to have a single contention point within the end system and reduce the constraints on the scheduler.

Virtual circuits are the units to which quality of service guarantees and priorities are given. Thus cells do not have an explicit priority field in them. MSNA does not dictate what qualities of service should be provided.

The lower layers of MSNA are concerned with cell transfer. MSNA uses an internetworking layer (analogous to IP) to provide end to end transfer of cells. The data link layer of MSNA is concerned with cell transfer on a link (where a link includes a simple topology LAN such as an Ethernet or Cambridge Fast Ring (CFR)[2]).

2.2 MSDL on Fairisle

The Multi Serivce Data Link layer (MSDL) is the network specific part of MSNA. In Fairisle, MSDL is concerned with how cells are transferred down a single link, either between a host and a switch or between two switches. Two MSDL entities transfer cells over an *MSDL association* which is identified by two virtual circuit identifiers, one for each direction. MSDL is also concerned with how associations are set up and how VCIs are allocated.

The format of a cell for the Fairisle network is shown in figure 1. As can be seen, there are four bytes of virtual circuit identifier (VCI), two bytes of framing and sequencing

[2]The CFR is a 50Mbit/s slotted ring

information and 48 bytes of data.

VCI	FAS	DATA
(4 bytes)	(2)	(48 bytes)

Figure 1: Fairisle Cell Format

As part of association setup, an MSDL entity allocates the VCI for cells it is going to receive. The VCI has no structure visible outside the allocating entity. The allocating entity (which is the only entity which will have to process cells with that VCI) may of course place any internal structure it wishes on the VCI in order to make the processing task simpler[3].

Per cell error detection and/or correction mechanisms are not defined by MSDL or higher layers. MSDL is designed to operate over a range of networks, and even within Fairisle it is required to operate over at least two significantly different transmission systems. Rather than define a generic mechanism within MSDL, each underlying transmission system implements a technique which is designed to deal with their individual error properties.

MSDL definitions and implementations also exist for the CFR, Ethernet, and the Cambridge Backbone Ring (CBR)[4] [Greaves 88].

2.3 MSNL

The Multi Service Network Layer (MSNL) concatenates a number of MSDL virtual circuits at *MSNL routers.* A series of concatenated associations which forms an end to end channel is an *MSNL liaison.*

A router maps an incoming VCI to an outgoing VCI and performs simple cell forwarding. This takes place at every Fairisle switch and at a MSNL router joining Fairisle to another MSDL based network – a CFR for example. This mapping is established at liaison set up

[3]For example by only using some of the bits to perform direct lookups, by using some bits to indicate priority, etc.

[4]The CBR is a 500Mbit/s slotted ring

time when the MSNL layer performs routing and establishes a series of associations based on the MSNL addresses of the correspondents, the state of the network and the requested quality of service. This connection set up mechanism is performed out of band.

MSNL only defines the purpose of a single bit in the in band information. When used, the *frame* bit delineates end to end frames, and is set in the last cell of a frame. No maximum size is defined for a frame, and it is conceivable that for some traffic classes the frame size will be a single cell (e.g. voice traffic).

The frame bit is set by the sender and will usually represent the unit of end to end recovery. The information is available at the receiver, but most importantly it can be used by the switch as part of the cell discard algorithm. During times of overload, when a switch discards a cell, it will preferentially discard cells arriving on the same virtual circuit up to and including the next cell with the frame bit set.

We envisage interworking with B-ISDN at the MSNL level by defining an appropriate MSDL instance over B-ISDN's null adaptation layer.

2.4 MSSAR

Two bytes of the cell are shown as allocated for sequencing and framing information. MSNL defines the purpose of the framing bit only, the remaining (15) bits are available on an end to end basis. However, we have defined a generic sequencing mechanism which uses this field to identify lost cells, and although not a requirement, we envisage all traffic classes using this service. While the use of different sequencing mechanisms for each traffic class could provide a small saving in bandwidth, a generic solution allows the sharing of both software and hardware components.

The Multi Service Segmentation and Reassembly (MSSAR) is one user of this framing and sequencing service, providing the further services of end to end error detection / correction (and encryption if required) on blocks of data.

3 Network Components

The Fairisle network is an experimental network. It purpose is to provide insight into the management of ATM networks. Thus the most important consideration is flexibility rather than speed or cost.

3.1 The Fairisle Switch

The Fairisle switch is a space division ATM fast packet switch. It is input buffered and blocking[5]. The current switch size is 16 ports. The switch can be broken into two types of component: switch fabric and port controllers.

Port controllers are analogous to line cards in a telephone exchange. Their main functions in the Fairisle switch are to map VCIs, manage queues, select priority, select routing tags, and deal with blocking in the fabric. Each port controller (there are 16 in each switch) is connected to an input and output of the switch fabric.

The fabric is a very regular interconnection network. It is the place where cells contend for bandwidth. The fabric is relatively straightforward to describe; we will do this first.

3.2 The Fairisle Switching Fabric

Functional Description

The main thrust of Fairisle is to experiment with networks. For this reason we have choosen to use a simple and easy to implement switch fabric, based almost entirely upon the Cambridge Fast Packet Switch [Newman 88]. A fabric based on a full crossbar would provide greater per line utilisation. However, using the same technology we can implement a somewhat faster version of our fabric and achieve approximatley the same throughput; this is possible as our switch design does not require us to run the fabric at the same rate as the transmission system.

The Fairisle switch fabric is a 16 by 16 (that is 16 inputs, 16 outputs) fabric built up from

[5]That is a cell may block another cell even though the cells are destined for different outputs

eight 4 by 4 crossbar switching elements arranged in a delta network as shown in figure 2. The delta network gives rise to the internal blocking. (Consider when port 0 and 1 both wish to send to ports 5 and 6 respectively. The cells both wish to use the internal link from element A to F; only one of them can.) Other switch sizes are permitted — in particular a non-blocking 4 by 4 switch can be built, or multipath 8 by 8.

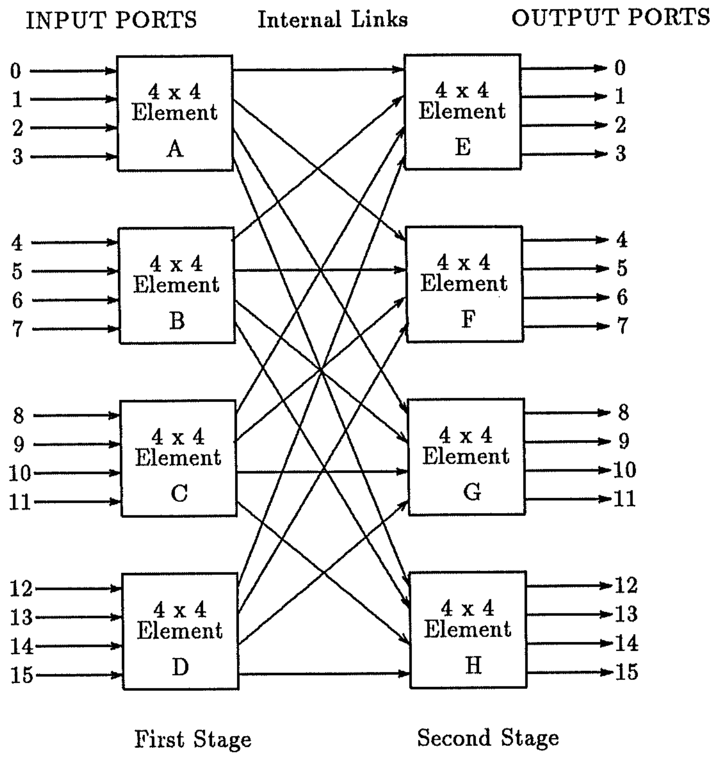

Figure 2: A 16 Port Delta Network

Each arrow represents 8 data signals and a returning acknowledgement signal

The fabric is self routing, that is, the cells have prepended to them *routing tags* which tell the fabric which links the cells should be sent along. The routing tags also contain priority information (in fact a single bit is used) which is examined when two cells contend for the same link. The fabric is a two stage delta so there are two routing tags. Each switching crossbar element has four outputs so two bits of routing information are required for each tag. Each stage strips off the routing (and priority) information which it uses as it processes the cell. In fact it is easier for the fabric to strip off whole bytes so routing tags are padded to a full byte. This is shown in figure 3.

Act (1 bit)	Pri (1 bit)	Route (2 bits)	Fill (4 bits)

Fabric Element Routing Tag

Act (1 bit)	Route (1 bit)	Fill (6 bits)

Output Port Controller Routing Tag

Stage 1 Tag (1 byte)	Stage 2 Tag (1 byte)	Output Tag (1 byte)	VCI (4 bytes)	FAS (2)	DATA (48 bytes)

Tagged Cell Entering Fabric

Stage 2 Tag (1 byte)	Output Tag (1 byte)	VCI (4 bytes)	FAS (2)	DATA (48 bytes)

Tagged Cell Entering Second Stage

Output Tag (1 byte)	VCI (4 bytes)	FAS (2)	DATA (48 bytes)

Tagged Cell Leaving Fabric

Figure 3: Tagged Cell Formats at Various Stages

Since the fabric (or indeed an output port controller) may chose not to accept a cell, an *acknowledgment* signal is provided. If the cell is blocked in the fabric, the input port on which the cell was injected will see a low on the acknowledgement line, otherwise it will see whatever the output port controller chooses to respond with — in fact it is a clear return data path from output to input.

Whenever two or more cells contend for the same output of a switching element, arbitration takes place. Arbitration is done for each of the four outputs of a switching element. Each output arbiter runs an independent round robin system with two levels of priority. The input links are numbered 0 to 3 and the last input to have a cell (of any priority) selected is remembered:

1. A high priority cell (one with the priority bit set) is always selected in preference to a low priority cell
2. If two or more cells of the same priority contend, the last input which was selected is used as the base. The next input above (with suitable wrap around) the last selected is the one to be selected.

The question of priority within the fabric is an interesting one. Just how many priority levels within the fabric are required will be a subject for experimentation.

Our switch fabric implementation does not represent a large amount of work. We are hopeful that self routing crossbars of a reasonable size will appear on the market in the near future. We see no reason why we could not include them in a later version of the switch.

Implementation

The fabric is implemented on a single board. The interconnection of switching elements into a delta network is entirely straight forward. The implementation problem is building a switching element.

Each element (a 4 by 4 crossbar) is built on a 4200 gate equivalent Xilinx programmable gate array. A link consists of eight unidirectional data signals and an acknowledgement signal travelling in the opposite direction. There are four input links and four output links making 72 signals for data and acknowledgement.

There are two other signals of importance — the byte clock and a frame start signal. The byte clock runs at 20 MHz giving a datarate of 160 Mbps. Frame start is used to synchronise cell submission to the fabric. The port controllers also use frame start for this purpose. The frame start pulse is in fact the only indication the fabric has about cell size; it is generated by a PAL on the fabric board.

3.3 The Fairisle Port Controller

Requirements

A Fairisle Port Controller is attached to an input switch port and corresponding output switch port, since it acts as both an input and output port controller. The port controller also interfaces to the transmission system[6].

Port controllers are MSNL routers. They receive cells from the transmission system and perform a lookup on cell VCIs in order to determine:

1. whether the VCI is valid,
2. which outgoing VCI(s) the cell should be forwarded on,
3. which output port on the switch the cell should be sent to,
4. which priority the cell should have through the fabric,
5. which queue the cell should be placed on
6. what discard and retry policies should be used on blocking

The switch is input buffered, that is, it is the input port controller which must buffer cells and deal with blocking in the fabric. Therefore as well as performing the lookup operation described above, the port controller must implement the retry and discard policies.

The output side is straightforward. There is a mechanism for routing cells either to the port controller itself (for management) or to the outgoing transmission system. Speed matching is provided by a simple FIFO technique, when the FIFO is logically full the output port blocks back through the fabric by keeping the acknowledgement signal unasserted.

The port controller can also gather statistics about cell flow. These can be use for policing and monitoring.

[6]Local links will use TAXI devices; longer links will be based on SONET and/or SDH trasnmission.

Implementation

The port controller is based on a simple high speed RISC processor, the ARM3[7]. This provides the main control function of processing cell headers and running the queueing algorithms. It is not, however, involved in moving cell data. A schematic of the port controller is shown in figure 4.

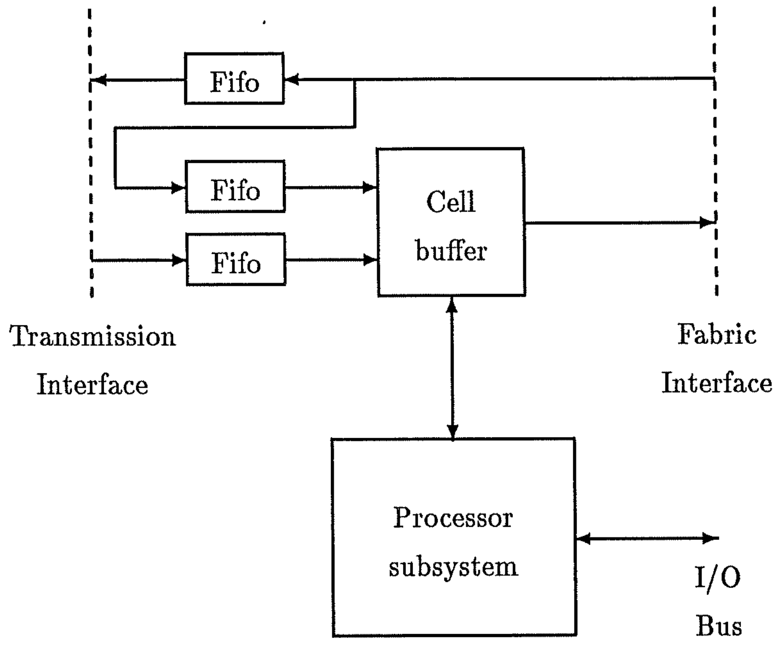

Figure 4: The Fairisle Port Controller

The main components are

1. the ARM3 processor system which includes a memory controller, an I/O controller, 1 Mbyte of DRAM, 128 Kbyte of PROM and an IO bus.
2. the cell buffer which is 128K bytes of SRAM (35 ns) and three 32 bit buffers.
3. a Xilinx device which triple ports the cell buffer and provides the glue to the processor, the fabric and the transmission system.
4. the fabric and clock interface — simply latches and terminations

[7]Designed and used by Acorn Computers Ltd, this component is sourced by VLSI Technologies Inc., CA.

5. the transmission interface comprised of FIFOs and terminations

The use of a processor provides an extremely flexible port controller. This means that various queueing algorithms, service qualities, policing techniques, and congestion control algorithms can be explored.

The Xilinx device is a 4200 gate equivalent device. However a pin compatible 6400 gate equivalent device is available should we wish to put more functionality (for example all functionality) in hardware.

Port Controller Software

The run time system for the port controller is a micro kernel called Wanda. This is a locally developed micro kernel, providing interrupt handling, address space management, threads, and interprocess communication mechanisms. (Wanda also runs on multiprocessor Fireflies and on 680x0 processors.)

In the port controller cells are placed on and removed from queues by interrupt service routines. (The number of instructions per cell at a line rate of 100 Mbps is on the order of 50.) The choice of which queues should be serviced is implemented by management algorithms at the MSNL level which run on a longer timescale than the individual cell.

In fact the implementation of MSNL within Wanda provides this cell forwarding function between any network devices implementing MSDL. The software which controls the Xilinx chip and SRAM on the port controller simply fit into this generic mechanism.

A version of Wanda is held in PROM (including the Xilinx configuration). On reset the software in the PROM attempts to obtain the latest version of the software from a boot server. We currently provide one port controller with an Ethernet interface to enable it to boot over the Ethernet. This then acts as an MSNL router, and allows the remainder of the port controllers on the network to communicate with the boot server.

3.4 Management Platform

The original intention was to have a switch controller which would be a processor attached to all port controllers via a management bus. However, the use of a processor in the port controller enables us to use either one or all of the port controllers as management entities. As the network management function is necessarily distributed, we adopt the latter approach as it provides us with a testbed with the largest degree of distribution possible.

4 Using The Network

4.1 Host Interface

Two configurations of a simple network interface for the VME bus are available, the choice of which to use is dependent on the function of the attached host. A simple programmed I/O interface provides cell level access to the network. A set of FIFOs interface directly between the VME bus and the transmission interface. As there is a considerable overhead in dealing with individual cells, such interfaces are only used when the machine is performing some dedicated task which leaves spare capacity on the processor.

Many hosts will wish to avoid handling cells and only deal in blocks, and in this case the VME board is combined with a port controller which provides the required segmentation and reassembly functions. The VME board attaches to the switch fabric interface of the port controller, so that, with simple modifications to the code in the port controller, incoming cells can be demultiplexed and reassembled into blocks before being passed on to the host.

4.2 Attachment to Other Networks

The port controller has an interface which conforms to Acorn Computer's expansion bus. Ethernet and CFR interfaces are readily available. Device drivers providing MSDL for these networks exist and hence each port controller can act as an MSNL router between

Fairisle and these networks. In particular interworking with the CFR provides access to the Pandora multimedia workstations [Hopper 90] as sources of real time traffic.

An MSNL router is also available between Fairisle and the CBR. This uses a dedicated 68030 VME system to perform the MSNL cell forwarding between an attached backbone ring station and the simple cell interface to Fairisle.

4.3 Continuous Media Devices

As with the interconnection of Fairisle and other networks, two choices are available for the attachment of continuous media devices to the network. Both the port controller I/O bus and VME systems, equiped with the simple cell interface, allow us to connect inexpensive devices, such as frame acquisition systems, to generate real continuous streams.

4.4 Traffic Generation

We consider it important not only to attach existing and real multimedia devices to the network, but also to be able to generate streams with a wide range of traffic characteristics and requirements. For example, we could perform experiments on the implications of a new viedo coding mechanism on the network, even before equipment implementing the mechanism is available.

Again the programmability of the port controllers (including the Xilinx device) mean that it is a relatively simple matter to generate streams of cells for virtually any statistical arrival process (including multi layer arrival processes). Thus port controllers are used as traffic generators, either to inject cells directly into the fabric or down a transmission line to a real port controller.

5 Conclusion

The architecture of the Fairisle network and some of the network components have been described. At the time of writing (March 91) the port controller and switch element are built and tested. We are currently awaiting PCBs for the switch fabric before commissioning the first switch. The VME bus interface is designed and now at the stage of PCB layout.

Projects within the laboratory will be making use of the Fairisle network for experiments and developments in multimedia applications. However, a key feature of the network components is their flexibility, and this also allows us perform a wide a range of network management and internetworking experiments.

References

[Greaves 88] The Cambridge Backbone Network.
DJ Greaves and A Hopper.
Eurpoean Fibre Optic Conference (EFOC / LAN 88), Amsterdam. June 1988.

[Hopper 90] Pandora - An experimental system for multimedia applications.
A Hopper.
Operating Systems Review Vol.24 No.2. April 1990.

[McAuley 89] Protocol Design for High Speed Networks.
DR McAuley.
PhD Dissertation, University of Cambridge Computer Laboratory, Technical Report 186. September 1989.

[Newman 88] A Fast Packet Switch for the Integrated Services Backbone Network.
P Newman.
IEEE Journal on Selected Areas in Communications, Vol 6, No 9, December 1988.

[Tennenhouse 89] Layered Multiplexing Considered Harmful.
DL Tennenhouse.
IFIP Workshop on Protocols for High-Speed Networks, Zurich. May 1989.

Fairisle Port Controller
Design and Ideas

Mark Hayter and Richard Black

March 1991

1 The Port Controller

The Fairisle Switch is based around a simple switch fabric with all the processing being done in the port controllers. The port controller consists of two parts. The first is a processing unit based around an ARM3 RISC processor. The second section consists of buffer memory and a DMA engine. Figure 1 shows an overview of the port controller, the lower half forming the processor section, and the upper the network section. There is a plug in transceiver card for connection of a fibre, alternatively for short links an 8 bit parallel connection can be made.

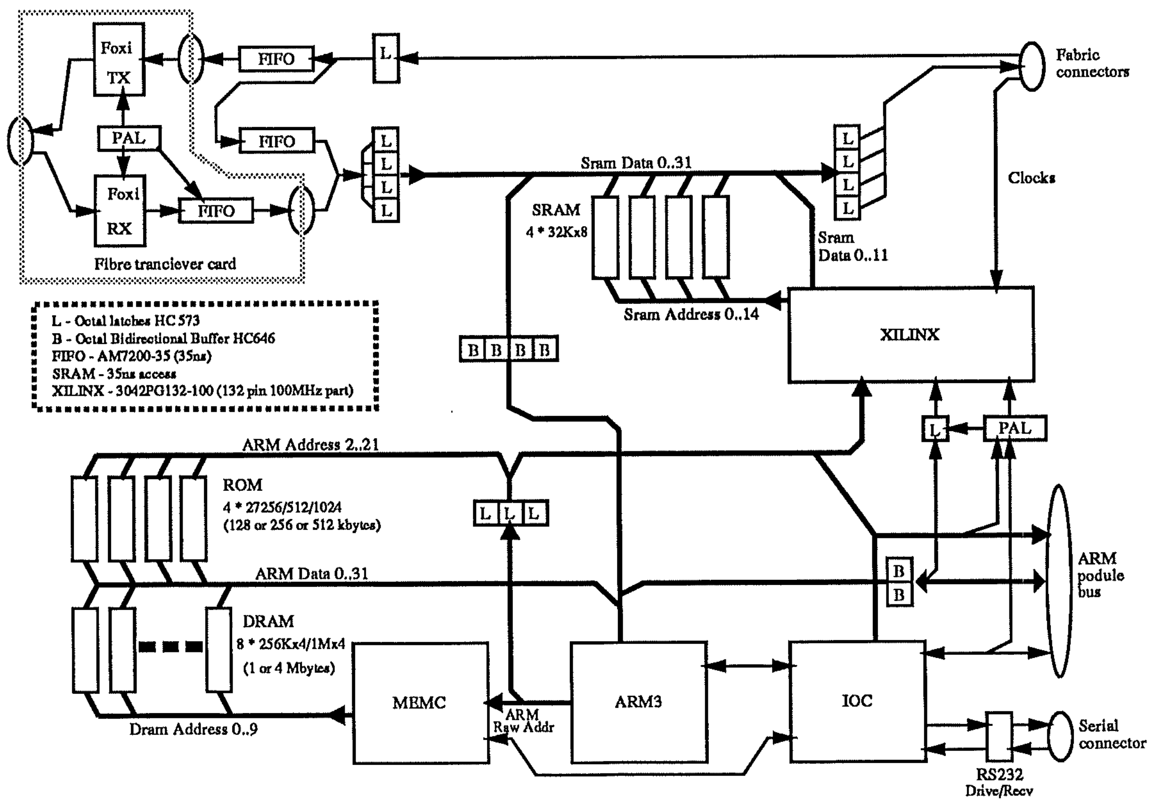

Figure 1: Port controller layout

The processing section of the port controller has an ARM3 processor and runs the Wanda micro-kernel. The current boards can have either 1 or 4 Mbytes of dynamic memory and 128, 256 or 512kbytes of ROM. The card includes an Acorn podule bus to allow the addition of other network interfaces. In addition to the standard ARM version of the Wanda kernel there will be a hand crafted interrupt routine to deal with the routing of cells. Cells are only buffered on the input to the fabric - the processor has no control of the fabric output.

The network section of the card contains 128kbytes of static memory, buffer fifos and the xilinx control chip. The static memory is used for cell buffering, and resides in the memory mapped I/O space of the processor. The xilinx arbitrates access to this memory between the processor, transmission to the fabric and reception from the network.

2 A Cell on the Port controller

The SRAM on the port controller is divided into cell buffers, which are written or read in a DMA fashion for incoming or outgoing data, and may be accessed as RAM by the processor. The cell buffer format is shown in table 1. The data from the network is bytes

Byte	Name	Comment
0	FabR1	Fabric 1st route byte
1	FabP1	Fabric 1st pad byte (always 0)
2	FabR2	Fabric 2nd route byte
3	FabP2	Fabric 2nd pad byte (always 0)
4	PortPad	Portc pad byte (always 0)
5	PortR	Portc route byte (bit 0 - set, bit 1 set/clear for loop/tx)
6-11	Header	VCI and SAR bytes
12-59	Data	User data bytes
60-63	Spare	Unused, only exist in port controller buffer

Table 1: Cell Buffer Format

6 to 59 of the cell buffer, the routing tag being added by the port controller. The full 60 bytes are sent into the fabric, which passes bytes 4 to 59 to the output port controller. The pad byte is removed and the route byte is replaced by a start of cell control signal for transmission on the network. The spare bytes are never used by the network section, so are free for internal use by the processor section. The route byte formats are given in tables 2 and 3. Note that since the routing bytes are discarded on the way through the fabric and port controller the spare bits cannot be used to send information. The fabric is being redesigned to remove the pad bytes, the port controller one is only present to word align data.

Bit	Name	Use
0	Active	Always set for a cell
1	Priority	Set for high priority cell
2-3	Route	Output requested for this cell
4-7	Spare	Unused

Table 2: Fabric Route byte

Bit	Name	Use
0	Active	Always set for a cell
1	Loop	set for loopback, clear to transmit to net
2-7	Spare	Unused

Table 3: Port Controller Route byte

3 The xilinx DMA controller

The xilinx chip, which controls the network part of the card, consists of five main sections: the processor interface, the SRAM interface, reception from the network, injection into the fabric and transmission to the network. It maintains two queues of buffer pointers into the SRAM. Buffers in the first queue are filled by incoming cells from the network, the processor then inserts a routing tag and adds the buffer to the second queue for injection into the fabric; it may be that the VCI of the cell is also changed at this stage. The processor is interrupted if either of the queues are not full of buffer pointers (indicating cell arrival or transmit availability), or a NACK is received from the fabric. Figure 2 shows the various parts of the DMA system and gives an outline of their interconnection. All the parts inside the grey border are on the xilinx device.

3.1 SRAM interface

The xilinx is the only device which drives the SRAM, arbitrating between its DMA use and the processor. The arbiter assigns top priority to the fabric transmission requests, next to network reception requests, and lowest to processor accesses. Access grants control a multiplexor which drives the SRAM address bus. Only part of the SRAM data bus is used directly by the xilinx, to load the buffer registers and output its internal status. Enables of SRAM data bus drivers are also controlled by the xilinx.

3.2 Injection to the fabric

The xilinx will start sending cells into the fabric if there are buffers in the transmit queue and the processor has strobed the RamRdGo line since reset or a fabric NACK. Transmission continues until the queue is empty, or a cell is NACKed by the fabric or destination port controller. If transmission stops because the queue becomes empty then it will restart as soon as a new item is put in the queue – strobing of the RamRdGo line

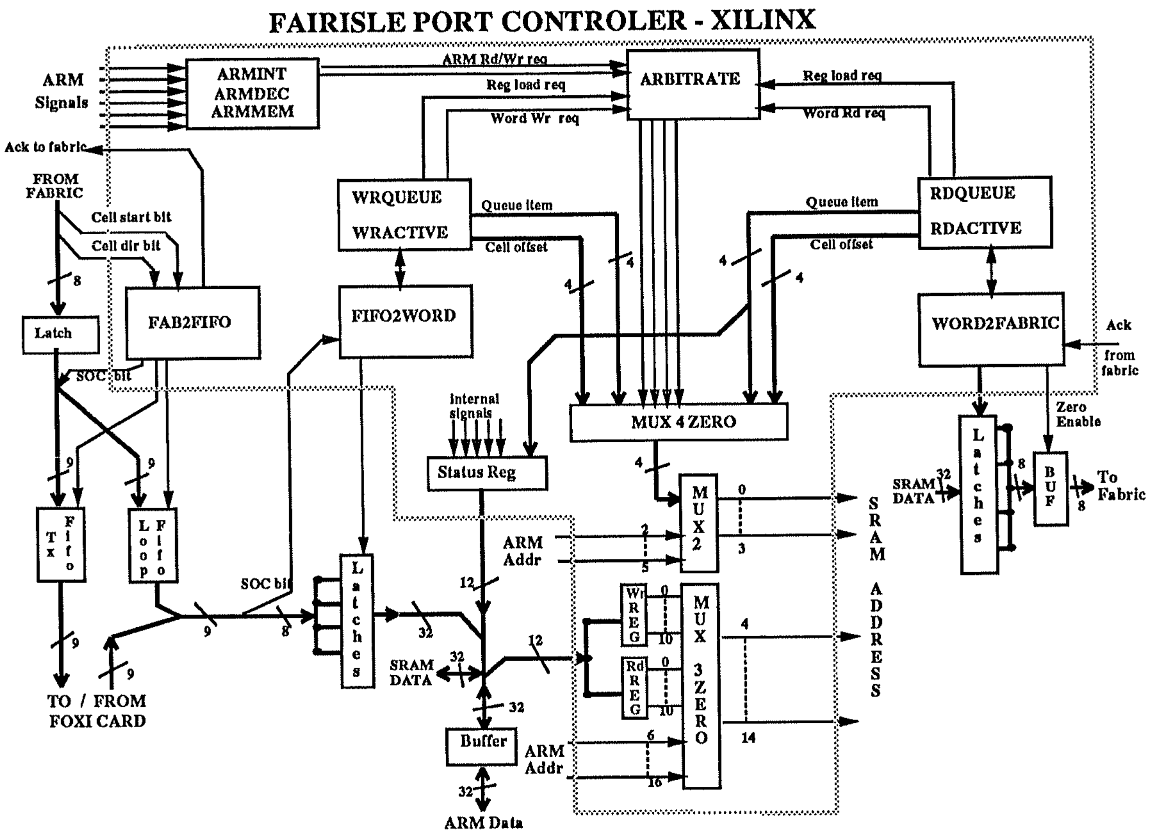

Figure 2: Port controller DMA system

is only required after NACK or xilinx reset.[1]

To send a cell the "word2fabric" section of the xilinx first loads the register RdREG from the read queue location in the SRAM. Then words are read from the SRAM, using the register value to form the top part of the address and the read active counter to supply the offest into the buffer. The word is loaded into the latches and clocked into the fabric a byte at a time. The next word is loaded in the same cycle as the 4th byte is output. Since the arbiter gives these memory reads top priority, this ensures the stream of bytes to the fabric is continuous. The fabric uses the bottom bit to mark the start of the cell, so an external buffer is used to force this bit to a zero when no cell is being transmitted.

The Ack signal from the fabric is monitored just after the header has been sent. If a NACK is seen at this point then transmission of this cell is stopped at once. If the cell buffer pointer has its retry bit clear or the cell has been retried already the transmission system is disabled. The processor can be interrupted by transmission stopping and can determine which cell was rejected (from the status register) then implement the correct retry strategy.

[1]It would be easy to change the hardware so that the line had to be strobed to restart after a queue empty state. Currently I can see no advantage to doing this.

3.3 Fabric output to network

Data from the fabric is moved under control of the xilinx "fab2fifo" section, into one of the two fifos. The cell arriving from the fabric has had the fabric routing information removed but still contains the port controller route byte. The route byte contains two bits that are used by the port controller. The cell Active bit is always a 1, and is used to indicate the start of the full cell in the same way as in the fabric. The loopback bit indicates whether the cell is destined for transmission or for looping back into the port controller.

Bytes arriving from the fabric are latched to provide a cycle delay for decoding the routing information. On recognition of a cell the loopback bit is used to indicate which fifo should be used for the duration of this cell. Bytes are then strobed into the fifo for the number of bytes in a cell. The fifo contains 256 nine bit words. The ninth bit is used as a start of cell marker and is set as the portc route byte is strobed into the fifo. This byte will be removed by the FOXI transmission card and replaced by a control code. The data associated with this byte is ignored by the transmission system.

The Half Full signal from the selected fifo is used to generate the Ack signal to the fabric. Thus as soon as the fifo becomes half full back pressure is applied. Since the transmitting port controller only senses the Ack signal for one byte time just after the header has been transmitted it is possible for almost an entire cell to be put in the fifo past its half full point.

3.4 Reception from network or loopback fifo

The FOXI reception card contains a fifo into which an incoming cell has been strobed. The start of cell control sequence will have been replaced by a byte with the SOC bit set. Thus the format of the data is the same in both the loopback fifo and the receive fifo. The data in this byte is currently not defined [2].

If either fifo is seen to contain data then it is selected for reading, if both contain data then they are read alternately. When a SOC bit is seen from one of the fifos it is selected. This also causes the WrREG to be loaded from the write queue location. Bytes are read from the selected fifo and assembled into words in the latches. When a word has been assembled it is written to the SRAM using the register buffer address and the offset suplied by the write active section. The write active starts with an offset of one word and the latch loading starts half way through the word. This leaves 6 bytes at the start of the buffer for the processor to insert the route tag. If a new SOC is seen part way through reading a cell then the first cell is discarded and the new one read into the same cell buffer, a status bit is set to indicate this happening.

3.5 The processor interface

The processor control signals are decoded and I/O accesses to the xilinx space (0x3180000 - 0x31FFFFF) are detected. The use of I/O space causes the access cycle to be asynchronous, allowing the xilinx to arbitrate the memory access. The SRAM is mapped into this area for 32 bit word accesses. The top address bits allow modifications to the access method. A basic access allows reading or writing of the memory. If the strobe bit is set then on a write operation one of the queue insertion lines will be strobed, bit 5 determining

[2]In most cells this byte is strobed into the portc route byte location in the cell buffer, one idea is that an identification be assigned to each type of interface so the port controller could use this byte to determine the characteristics of the interface.

which. The final access method is a read with the status bit set, this causes a fake memory access with the data being supplied from the xilinx status register. Addresses 0x3180000 and 0x31C0000 are ARM immediate constants and so the most used access methods are placed at these addresses. A byte write usually results in all 4 bytes in the addressed word being set to the value given. Since an asynchronous memory access is used it will take at least 2 cycles of the ARM REF8M 8MHz clock to perform the access.

Start	End	Use
0x3180000	0x319FFFF	Read / Write access to SRAM
0x31A0000	0x31BFFFF	Read xilinx status word, write SRAM
0x31C0000	0x31C001F	Read SRAM / Write output queue with strobe
0x31C0020	0x31C003F	Read SRAM / Write input queue with strobe
0x31C0040	0x31DFFFF	Read SRAM / Write SRAM and strobe (bit 5 -> in/out)
0x31E0000	0x31FFFFF	Read status, write with strobe

Table 4: Memory map

The memory map is shown in table 4. The first 8 words of the SRAM are the output queue, the next 8 the input queue. Both are circular, with the processor inserting buffer pointers, and the xilinx consuming them. It is the responsibility of the processor to ensure a new buffer pointer is stored in the correct memory location to be added as the tail of the

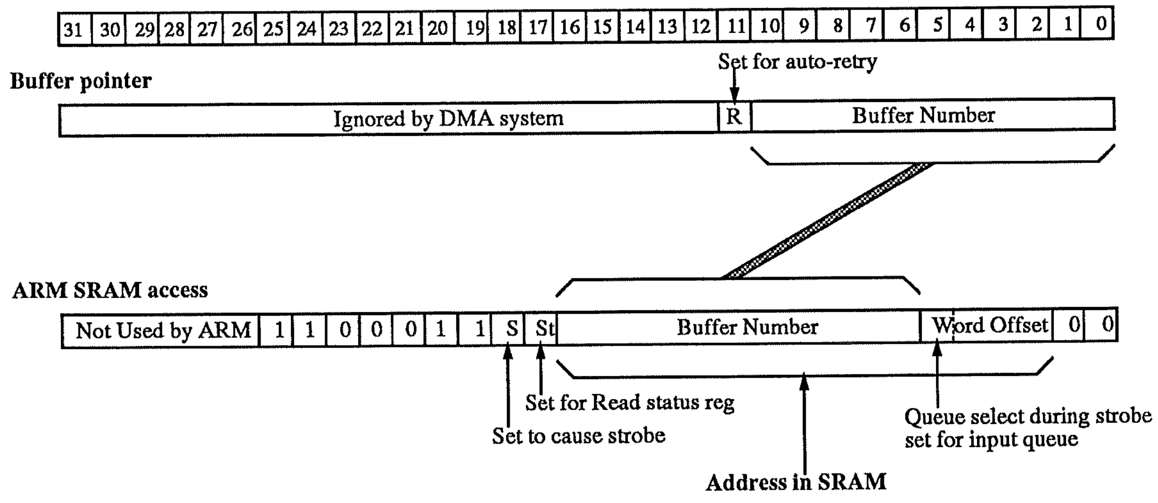

Figure 3: Buffer pointer and memory addresses

queue. There is no protection against the processor strobing pointers into a full queue - if it does it will overwrite an existing buffer pointer but the internal tail will **not** advance. The processor can cause a queue strobe signal by writing anywhere in the memory with the appropriate high order bits set.

The SRAM is divided into cell buffers. The zeroth buffer contains the queues, the rest are available for cell DMA. The mapping from buffer pointer to buffer address is simply a left shift six bits of the bottom eleven bits of the pointer. The higher order bits of the pointer are not used by the DMA system, so may be used by the control software. Note that it is the buffer pointer, and not the buffer address, that is stored in the queue. Figure

3 shows the buffer pointer value and an ARM memory address and their relationship to the SRAM address (which accesses a long word).

The buffer pointer includes a retry bit. If this is set then the DMA system will do one automatic retry should the cell be NACKed on the first attempt. If the retry fails then transmission stops and the processor is interrupted. If the retry bit is clear then the system stops on the first NACK.[3] The ARM memory address shows the bits that are decoded to cause a status register read, or a strobe with write.

4 Status and Control

The xilinx status can be found from its internal status register, and from signals passed to the processor section I/O controller (IOC). The lines on the IOC can be set to cause a high priority, low latency, interrupt (a FIQ) to the processor. Tables 5[4], 10 and 7 describe the status signals.

bit	name	use
0-2	RdTailAdr	Buffer queue number for cell being or about to be sent. Only useful if sending has stopped, when it contains the queue number for the next cell to be sent.
3	Spare	Always zero
4	RamRdReady	Set if there are buffers to send. Clear if the queue is empty.
5	RamRdFull	Set if output queue is full
6	RamRdOKbar	Set if not sending
7	RamRdRun	Set if output system is ok, cleared by reset or fabric NACK.
8	SICseen	Set if Start of cell mark has been seen in a cell, Cleared when status register is read
9	StrayBytes	Set if bytes seen from the fifo which were not part of a cell, Cleared when status register is read
10-11	Spare	Not used - always zero
12-31	-	Undefined. May read as any value.

Table 5: xilinx status register

The xilinx status register is mainly intended so that the processor can determine why sending to the fabric stopped. If the system is not sending (and has been) the RamRdRun bit indicates if it stopped because the buffer queue was emptied or because of a fabric NACK. The RdTailAdr gives the RdQueue index into the output queue. This index is for the next cell to be transmitted, so if transmission was stopped by a NACK it is the previous queue entry which contains the buffer pointer for the rejected cell. Since the index value is incremented part way through transmission of a cell it only gives useful information if the system is not sending. Some of the data bits are cleared after the status

[3] I have no idea if this is useful, but it is there to be tried. The number of retries could be increased easily.

[4] The spare bits may be used to bring out other internal signals if required. The undefined bits are not available.

register is read, these all latch unexpected conditions (such as short cells) seen by the xilinx [5].

The IOC control lines are used to control aspects of the xilinx chip, as detailed in table 6.

Bit	Name	use
0-1	I2C[0-1]	Podule Bus control
2	RamRdGobar	Strobe low to start xilinx sending cells
3	FifoReset	High to reset the Fifos, low to enable
4	Led1	Low to turn on red LED
5	Led2	Low to turn on yellow LED

Table 6: IOC Control lines

The control register lines have pull up resistors on them so on reset, when they are undriven, they will go high. Thus a card reset will cause the fifos to be reset, and the strobe line to be deasserted. The RamRdGobar line should be strobed low to start cells being sent after the xilinx is reset or a NACK has been received, it should be set high (ie undriven) again immediately. Extra strobes of the RamRdGobar line may cause NACKs to be missed (since they could be cleared before the processor has noticed them). The top two bits of the IOC control register must always be set when the register is written.

5 Programming the xilinx chip on the port controller

The xilinx gate array on the port controller is configured in peripheral mode (M0=M2=+5V M1=gnd) for easy programming under software control. There are three status lines which come from the xilinx chip to the IOC. These are INIT, HDC and RDY/BUSYbar which appear in IOC status register B[6] as noted in table 7.

Bit	name	use
0	PFIQbar	Podule FIQ
1	XiInitbar	Xilinx Init line
2	XiHdc	Xilinx high during configuration
3	XiRdy	Xilinx ready
4	LIRQ	Transputer link IRQ
5	PIRQ	Podule bus IRQ
6	STx	KART Tx Data empty
7	SRx	KART Rx Data full

Table 7: IOC Interrupt status register B

In addition to the data there are three control lines which affect the xilinx chip. These control lines are manipulated by the processor through a simple PAL. These lines are the

[5]There is a problem routing these signals on the chip so they may be removed.

[6]These interrupts are not enabled so do not appear in the request register

WriteStobe (WRTbar) active low line, a low to high transition on which causes the current value on the data lines to be read in by the chip. The other lines are the reprogram line which can be used to cause an already programmed chip to be wiped, and the reset line which is connected to global reset on the chip. In addition the PAL controls the enable line of a buffer between the IOC data bus lines and the xilinx data lines.

On power on reset or board reset the PAL asserts and holds the reset signal and deasserts all other signals. This ensures there are no board conflicts and prevents the xilinx chip from operating until the software chooses to bring it out of the reset state.

The PAL operations are controlled by the address bus only, appearing in the IOC space at device 1, see table 9, and are activated by a write to the given address. The pal equations may be found in */usr/groups/fairisle/designs/portc/revisions/PALREV.MAP.*

Name	Assert	Deassert
WriteData	0x3210000	0x3210004
DataEnable	0x3210008	0x321000C
Reprogram	0x3210010	0x3210014
Reset	0x3210018	0x321001C

Table 8: PAL access addresses

The first thing that the software does is to being the chip out of reset and wait for the INIT signal to go inactive. This indicates that the xilinx chip has completed its internal clearing operation. (If the chip has currently got a configuration in it then this signal will not have gone active and the chip will be running with the old configuration).

Then the HDC (High During Configuration) signal is tested. This signal is not valid during reset or initialisation phases but otherwise indicates if the chip is ready to receive a configuration. This signal will be high at this point iff there is no configuration in the chip. If this signal is low then the chip must have its current data removed, this is performed by asserting the reprogram line for a short period and waiting for the INIT line to go inactive. The xilinx allows for the HDC pin to become a user I/O pin after configuration but this is not permitted on the port controller. This line has a pull down resistor so can be left unused in the xilinx design.

In either case the chip is now ready to be programmed. This is performed by loading the configuration data bytewise, with the BUSY line being checked to ensure handshaking. Bytes are written using four operations.

1. The Data connect line is asserted. This links the xilinx data lines with the IOC data lines. The IOC data lines are isolated from the processor data lines except when the processor issues an address in the I/O space. Thus opcode fetches do not reach the xilinx data pins.

2. The Write line is asserted; the value written being the data byte required to be sent to the chip. At the end of this cycle the data is floating on the linked I/O - xilinx bus.

3. The write line is deasserted causing the xilinx to latch the data. Again the byte written to the address is the data byte for the chip. This ensures that the IOC buffers enforce rather than destroy the required data floating on the capacitance of the I/O bus from the previous cycle.

4. The data connect line is deasserted, decoupling the busses and permitting the processor to read the IOC register and wait for the handshaking signal.

It should be noted that these operations are extremely sensitive not only to delay but also to potential accesses to the I/O space. For this reason these operations should be performed with interrupts disabled. Once all the data has been written the HDC line should be checked for inactivity to ensure that the chip has accepted the data and assumed the configuration. After configuration strobing the reset line will reset the xilinx device, clearing all the internal flipflops. If this is done the fifos should also be reset using the IOC control line (see table 6).

6 The network interface

The port controller has a parallel network interface. On to this may be connected a transceiver for a fibre, or it may be used to directly connect devices. The interface is byte wide, and is asynchronous. Two connectors are used, one for transmission and one for reception, as detailed in table 9.

Pin	Signal Name	Transmit Direction	Receive Direction	Use
Even 2-24	Gnd			Ground
Odd 1-15	Data[0-7]	Out	In	Bytewide data lines
17	SOC	Out	In	Set for start of cell byte
19	Readbar	In	Out	Strobe low to read next byte
21	Emptybar	Out	In	Clear if there is no data
23	N/C			Not used
25,26	(VCC)	LK5	LK4	Power if link is made

Table 9: Port Controller network interface

Data is extracted a byte at a time on the transmit side by strobing the Readbar line low. The start of a cell is marked by having the SOC bit set, the data which accompanies this is undefined. Following the SOC are the 54 bytes of the cell, being bytes 6-59 of the cell buffer described in table 1. If there is no data available then the Emptybar line will go low, and reads are undefined. The transmission system should read this stream of bytes and convert them into the from required by the link. The reception system should convert the form on the link back into the SOC and bytes form [7], which it should supply bytewise when the read request is strobed on the receive side. The port controller expects to receive in the same form as it sent, so the simplest link may be provided by cross connecting the transmit and receive connections of two port controllers using ribbon cable. Alternatively a card is being made which will use the FOXI chips to allow a fibre link to be made. Another possibility is for a parallel connection to be made to the interface card of a host machine. The timing details for the reads and empty signal are as for a Am7200-35 CMOS fifo, with a 45nS read cycle, 35ns read access and empty going low 30ns after the read cycle of the last byte is initiated.

[7]Possibly the data of the SOC byte will contain an interface type identifier.

7 The interrupt handler

As mentioned in section 1 the ARM processing part of the port controller includes a hand crafted interrupt routine known as a FIQ. This is a low latency interrupt which may respond with a minimum delay of 170ns.[8] In order to further improve the latency of fiq interrupt response the ARM provides six banked registers which are only accessible in fiq mode and which may be used both to preserve state from one fiq to the next and to avoid having to save registers before use.

bit	Name	Use
0	FH[0]	Set on input queue not full
1	FH[1]	Set on output queue not full
2	FLbar	Set if output has stopped
3	C[3]	State of fifo reset signal
4	C[4]	State of LED1 (red)
5	C[5]	State of LED2 (yellow)
6	IL0	Podule FIQ line
7	-	Always set

Table 10: IOC FIQ Status register

The fiq interrupt can be generated by three separate lines from the xilinx device as described in table 10. The status of these lines is always software readable and they may be individually masked.

The fiq is responsible for the flow of data through the port controller using the xilinx device as a fancy DMA engine. The fiq's tasks can be divided as follows.

- It must transfer "forward" received cells to the transmit queue for injection to the switch fabric, inserting the routing tag and performing VCI remapping.
- It must keep the xilinx receive queue full to avoid packet loss.
- It must bring management and connection setup cells to the attention of the higher level Wanda system.
- It must merge management and connection setup cells being sent by the Wanda code with cells being forwarded.
- It must maintain statistics on forwarded cells to permit the Wanda code to perform policing and quality of service operations.
- It must restart injection under circumstances of Fabric NACK.

For testing the only fiq interrupt which is enabled is the receive queue not full interrupt. On receipt of this interrupt the fiq reads from the xilinx receive queue entry of the cell that arrived, converting the buffer number to the address. One byte of the VCI of the arrived cell is retrieved for inspection. If the value is greater than 15 then the cell is posted

[8]Due to the asynchronous nature of some ARM I/O accesses there is no obvious bound on the maximum value.

to Wanda by writing the buffer pointer into a global variable and generating an "always IRQ" (note a potential overrun could occur here if another such cell arrives before the Wanda system has processed the first.) If the byte was less than 16 then the Fabric, Port Controller, VCI and SAR fields are replaced by entries in an indexed table. The buffer number is then written into the transmit queue at the tail position as indicated by the xilinx status register. No check is made for this queue being full and no interlock with Wanda transmissions is provided. Finally in both cases the receive queue is filled up with the next numerical buffer number (this is held in a state register.) Again no check is made that this buffer is actually free.

This implementation is obviously very naive but gives an idea of the requirements and is good for testing the hardware.

7.1 Timing

The port controller's design speeds are 100Mbits/s on the network interface and 20Mbytes/s on the fabric interface. The ARM processor has internal (FCLK) cycles clocked at 30MHz. Cells arriving from the network contain 55 bytes, cells to the fabric 60 bytes with one frame pulse every 64 clock periods. Thus at saturation there could be one receive interrupt every 132 FCLK, and one transmit every 96 FCLK. This is not counting NACKs (which may occur on 60% of cells at saturation), but assuming that more is being injected than is being received (which is unlikely). In the current experimental phase without real sources of traffic or multiple port controllers it is difficult to determine the frequency and conditions under which a NACK interrupt is likely to occur.

The ARM can perform one cycle in an FCLK period provided the data is in the cache. Non-cache accesses are slower, an access to the sram taking at least two REF8M cycles, more than 8 FCLKs [9]. Time is therefore of the essence and it seems likely that a statistical, if not absolute, target should be 60 FCLKs for an interrupt.

It is expected that under normal operation the fabric will be able to sink all the traffic arriving from the network, and therefore that most of the time one does not want the transmit queue not full interrupt enabled since there will be no appropriate action to take. Thus the perceived situation is that the tx interrupt is not enabled but the rx interrupt is. In this mode, when an interrupt is taken and a cell needs to be forwarded, there will be space in the transmit queue so the cell can immediately be inserted in it. Although this is likely to be the most common state its operation is logically a subset of the general case where queueing is required. So it is the general cases that will be described. The normal "idling" case will be returned to at the end.

7.2 Queues

Queues of cells are required both for cells that are being forwarded into the fabric and those which are being delivered to or transmitted by Wanda. Ideally the fiq code should not have to take different action depending on what type of consumer a particular queue has. For time reasons it is very important that when Wanda is accessing these queues it is not necessary to lock out or disable fiq interrupts. The structure of a queue must therefore allow for Wanda to have a queue which it is accessing asynchronously updated.

A queue consists of a head pointer and a tail pointer. A map is kept with one integer per cell buffer and the chaining of the cells is done implicitly in this map. Cell buffer zero

[9] It may prove possible to run the REF8M clock at 10 or even 12MHz rather than the usual 8Mhz

can never be used because it holds the xilinx queues so it is possible to use zero as its traditional meaning of "null". It is also intended to reserve two additional cell buffers. One to read cells into under overload conditions where multiple cells may overwrite one another. This is preferable to not giving a buffer pointer to the xilinx as otherwise the fifos would fill up and there would be short cell problems, and also problems for the multicast ring (if present). The other would contain zeros (no start of cell bit) and could be used under panic conditions to avoid a xilinx transmit interrupt without consuming fabric or destination resources.

In order to allow the asynchronous updating of the queues the head pointer must only be updated by the consumer and the tail pointer by the producer. This in turn requires there to always be a valid pointer in these fields ie. that there is always one buffer associated with a given queue.

The entry in the map for a particular buffer contains a pointer to the next cell in this queue. A zero indicates the end of the queue. [10] Since there must always be a buffer in a queue even if it is empty a buffer whose next pointer is zero does not contain valid data but is the one into which the next set of valid data should be placed. Unfortunately this slightly complicates checking if the queue is empty.

Since the fiq is not-interruptible it may update the structures in whatever way it sees fit. However when Wanda, which is interruptible, accesses the queues it must obey the following procedures.

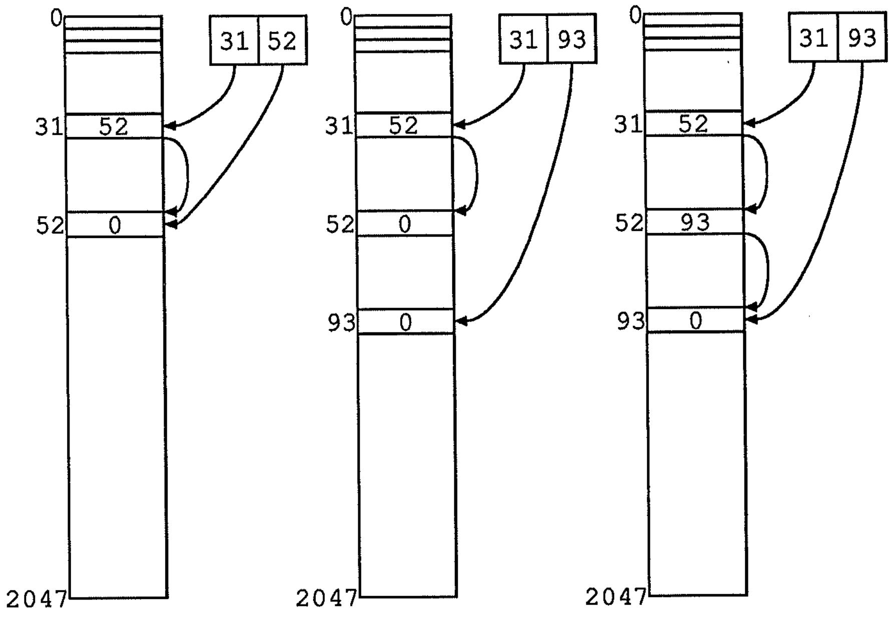

Figure 4: Inserting a cell buffer into a queue.

7.2.1 Insertion

The tail pointer of the queue is read. That buffer is then filled with the appropriate data. A new end of queue cell buffer must be allocated and its next pointer set to zero. The previous tail cell's tail pointer is set to the newly allocated buffer. This is a single

[10]Buffer 0 is not valid anyway since it holds the xilinx queues.

atomic write operation which validates the new data and makes it possible to be consumed. Finally the queue tail pointer is set to the new tail value.

7.2.2 Removal

The head pointer of the queue is read yielding a buffer number. If that buffer's tail pointer is zero then the queue is empty. Otherwise the data may be consumed. The next field of that cell is written to the queue head pointer and the buffer may later be freed.

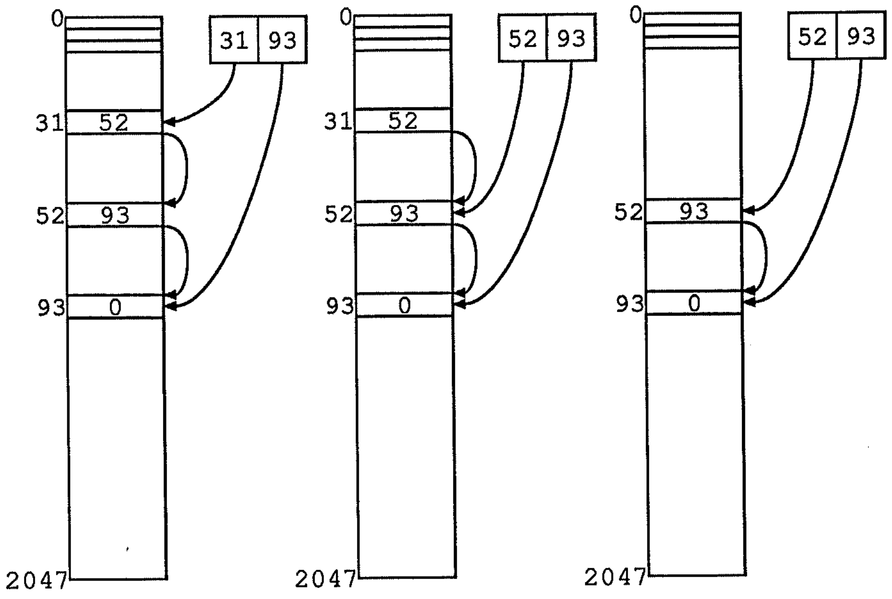

Figure 5: Removing a cell buffer from a queue.

7.2.3 Free List

The problem with the free list which is worse than the queues is that both the fiq and Wanda will want to update at both ends. There are three possible implementations.

1. Have a conventional queue as above but make Wanda prepare all the things in advance and then briefly disable the fiq interrupt and do the operation. The biggest disadvantage with this is that it requires the fiq to be disabled.

2. Have a doubly linked list with success checking by Wanda. This involves a conventional doubly linked list with the fiq working at one end and Wanda working at the other, with the rule as for the queues that there is always at least one buffer in the list. Again the fiq may perform whatever operations it requires and Wanda must obey special conditions when consuming a cell. The problem occurs when there are only two cells left and as Wanda allocates one, the fiq allocates the other. Referring to figure 6 the first line shows the positions of the data structures as Wanda removes a cell ("B") with the fiq idle. However if the fiq removes cell "A" between Wanda following the previous pointer from "B" and clearing the pointer from "A" to "B" then it would be the case that both cells would be allocated and both end pointers would be incorrect. To avoid this Wanda must check the "B" to "A" pointer

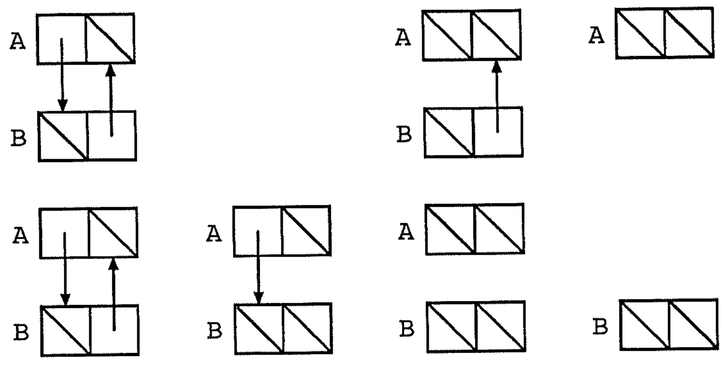

Figure 6: Arangement of the free list.

after clearing the "A" to "B" pointer to check that it still points to "A". If it does then everything is OK, if it is null then there are no free cells and if it has changed then the fiq must have both removed and added a cell and the procedure should be repeated. This algorithm now works except if the fiq allocates "A", deallocates another (say "C"), and then deallocates "A" again in the critical period. This is unlikely to happen.

3. Have an circular array of size 2048 with a single region of entries which contain the available cell buffers. A value of zero indicates no free buffer known for this slot in the array. Again the fiq and Wanda work from different ends. In this case Wanda uses is the ARM's indivisible swap operation to allocate a cell. If the value pointed to by Wanda's pointer is zero then the queue is empty. Otherwise Wanda swaps in a zero and gets the current value. If that value is zero then the list has just become empty. Otherwise the value obtained is the number of a free buffer and the pointer may be increased. This is the preferred method due to the much smaller overhead required from the fiq to allocate a cell buffer.

7.2.4 Size

If required the size of the queue would be kept as the difference between counters of the number of cells inserted and the number of cells removed. This keeps atomic update but means that the size of the queue may not necessarily be accurately known.

7.3 Arriving Cells

Arriving cells have their VCI used as an index to a table. This table shall contain at least three control flags as follows.

1. An indication whether the cell should be discarded immediately or not. This is likely to be used for cells on the "myaddress" type VCI after startup or to discard cells on chosen VCIs when running short of buffer space.

2. An IRQ request. If this is set then an arrival of a cell on this VCI will be triggered to Wanda by posting a conventional interrupt.

3. Mapping required. Indicates whether the VCI, multicast, portc and fabric routing information of the cell should be replaced with data in other fields in the VCI table.

This is likely to be that case except for cells on connection setup VCIs and other such VCIs which are consumed by Wanda.

Obviously not all combination of these flags are necessarily meaningful.

Current experience with the CFR gateway indicates, albeit with mainly booting traffic, that about 1% of traffic only requires connection setup type higher level processing. Thus the typical case will be mapping and no IRQ.

It is also possible that the VCI table will include a counter of the number of cells received on this VCI which would be increased when one arrived.

If the cell is not to be discarded then the table should include a handle for the queue into which the cell should be inserted.

After this has been performed the xilinx receive queue must be refilled. In the case of a discarded cell the same buffer number may be re-used. Otherwise a buffer must be allocated from the free list.

7.4 Departing cells

When a transmit space available interrupt occurs the cell buffer must be added to the free list. Then a new cell to be transmitted must be found. This is likely to be done with some sort of pool of queue references which Wanda has decided require scheduling for transmission. It is not yet clear how this will be performed. When a queue is chosen a cell will be removed from it as described above and inserted in the xilinx queue.

A possible method for finding a queue is to have a chain of eight queue pointers where each queue is associated with a xilinx queue entry ie. If the next xilinx insertion point is n then a cell is inserted from queue n. It is then the responsibility of the Wanda scheduling software to update these pointers as appropriate.

Cells from Wanda requiring to be merged into the injection stream would be placed in some appropriate queue which Wanda would then schedule for transmission when appropriate.

7.5 Blocked cells

The Fabric generates a NACK whenever there has been contention, either for an output port or due to blocking within the fabric, and this port controller's cell has lost. A NACK is also generated by the destination port controller if its fifo is becoming full. On seeing a NACK the xilinx (possibly after one automatic retry) generates a fiq interrupt and stops transmission. At this point there remains about three quarters of a frame cycle left in which to prepare for restarting the xilinx chip. Failing this deadline will result in a frame being missed. This interrupt is thus even more time critical than usual. The tail of the transmit queue is incremented as usual at the same time as the NACK occurs positioning to the next cell. The position of the queue before and immediately after the NACK can be seen in figure 7.

Care must be taken if both a transmit interrupt and a NACK interrupt are pending since there may have been other successfully sent cells whose interrupts were not handled before the cell that was nacked.

There are a number of things that could be done to reschedule a nacked cell.

1. Retry the same cell. This is the simplest system to envisage requiring each buffer number in the queue to be shifted along. Unfortunately this mechanism has poor sta-

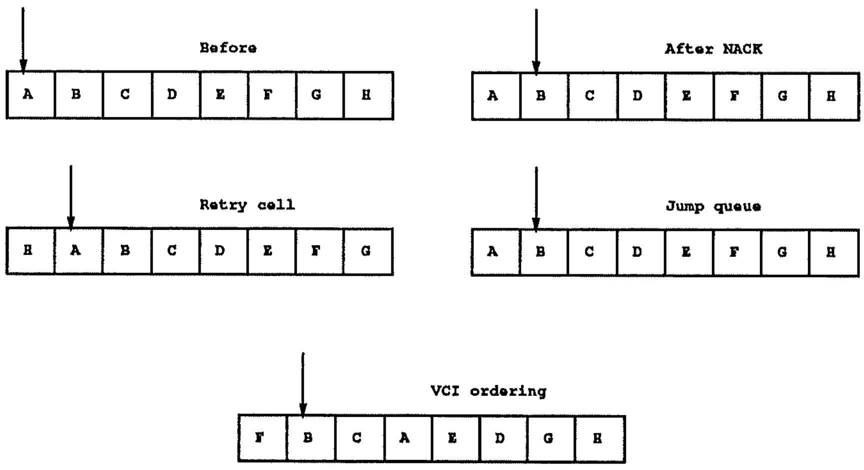

Figure 7: Fabric injection queues affected by NACK

tistical properties because one heavily accessed output port would delay all circuits. This is known as head of the line blocking.

2. Allow other cells in the xilinx queue to jump ahead of the NACKed one. This requires no moving of the queue [11] but suffers from a serious problem. If any of the cells which jump past have the same VCI then cell reordering will have taken place. This must be avoided by ensuring that there is never more than one cell on any VCI in the xilinx queue. Unfortunately this leads to cells waiting for transmission with empty slots in the queue and is also difficult to manage.

3. VCI reordering is essentially the same as the previous case except that on a NACK the queue is shuffled / sorted, eliminating the need for restricting which cells may be placed in the queue, though at substantial cost on a NACK. [12] In fig 7 for example if cells A,D and F were of the same VC, then the queue would have to be reordered as shown.

4. Another possibility is a variation of VCI reordering where the sort is done on the fabric routing information rather than the VCI. This is likely to be a little cheaper, but may or may not be statistically preferable to VCI reordering.

If the ideas of bundling (or grouping) of VCIs are used then it is VCIs in the same bundle which must not be reordered, so the size of the problem is increased. In this case the shuffle may need to be done on some other parameter, which is computed at connection setup time.

The xilinx control chip can do one automatic retry. This will suffer the same head of line blocking as (1) above, but used in conjunction with one of the other methods it may prove useful. It would also be possible for the hardware to generate an interrupt to indicate that it was about to do an automatic retry. The software could use this to start working out what to do if the retry fails. If software recovery takes longer than the critical time this might prove the best method, since the only loss is processor cycles should the retry succeed.

[11] If the queue is not full then the NACKed cell must be put at the end

[12] However the xilinx chip could be re-started as soon as the first cell of this re-ordering was completed.

7.6 The Idle state

As mentioned earlier it is expected that most of the time the port controller will not be queuing but idling. In this state the transmit queue available interrupt must be disabled, but the receive and nack interrupts must remain enabled. It may prove worthwhile to vector special code for this state instead of the general case. When a cell arrives, if it is to be forwarded then a check is made on the status of the transmit interrupt and if it is pending then the cell can be written into the xilinx transmit queue immediately without passing through any queue structures. If there is no such available slot then the fiq will queue the cell and switch to normal mode. If Wanda wishes to transmit in this mode then it will enable the normal mode, the fiq will transmit its cells and then automatically return to idling mode.

Fairisle VME Interface
— DRAFT —

Reto Beeler

March 1991

1 Introduction

The Fairlisle VME Interface is a half-duplex single channel VME slave inter- face for the Fairisle ATM transmission system. It supports D32 transfers (both A24 and A32), and it must operate under control of an external CPU or DMA bus master station.

2 General Overview

The Fairisle VME Interface is subdivided in a bytewide Fairisle ATM Interface (1),a MUX/DeMUX unit (2), VME access and VME interrupter logic (3), an address decoder (4), an Interface Status Register (ISR) (5) and an Interface Control Register (ICR) (6). (references to figure 1)

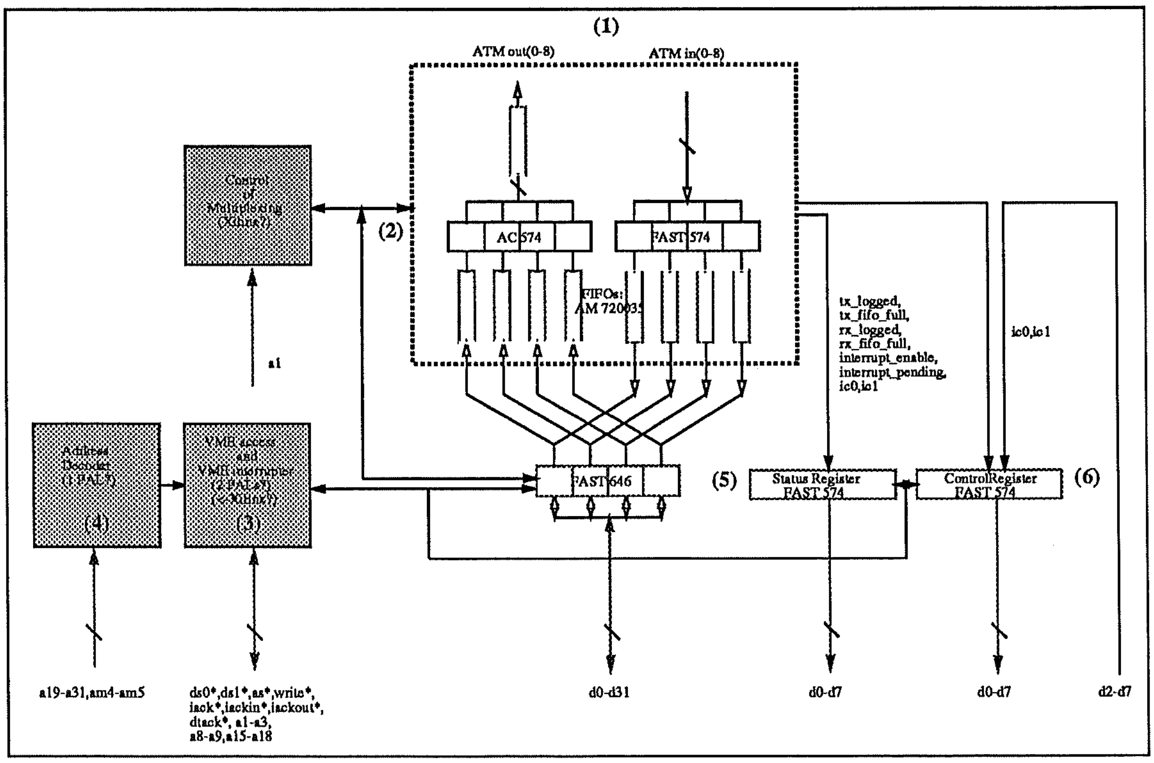
(1)
ATM out(0-8)
ATM in(0-8)
Control of Multiplexing (Xilinx?)
(2)
AC574
FAST 574
FIFOs: AM 72003
a1
tx_logged, tx_fifo_full, rx_logged, rx_fifo_full, interrupt_enable, interrupt_pending, ic0,ic1
ic0,ic1
Address Decoder (1 PAL?) (4)
VME access and VME interrupter (2 PALs?) (<Xilinx?) (3)
FAST 646
(5)
Status Register FAST 574
ControlRegister FAST 574
(6)
a19-a31,am4-am5
ds0*,ds1*,as*,write*, iack*,iackin*,iackout*, dtack*, a1-a3, a8-a9,a15-a18
d0-d31
d0-d7
d0-d7
d2-d7

Figure 1: General Overview

3 Addressing

3.1 Links

The current implementation of the address decoder allows for two alternative base addresses of the VME Interface Board: FDE0.0000 or FDF0.000. The base address is chosen by a link. A second link allows to select A24 transfers. Other links allow to set the VME interrupt level. (see Chapter 6)

Link	Link Function	Open	Closed
1	Base Address	FDE0.0000	FDF0.0000
2	Address Width	VME A32	VME A24
3-5	Interrupt Level (binary)	One	Zero
6-12	Interrupt Level (unary)	Off	Selected

Table 1: Link Setting Options

3.2 Device Addresses

The address space of the VME (FDE0.XXXX or FDF0.XXXX) is used by five devices with a distinct device address. These devices are the Interface Status Register (ISR), the Interface Control Register (ICR), the Transmit FIFO [1], the Receive FIFO [2] and a software reset address.

Device	Address	Operations
ISR	FDE0.0000	read only D8/D16/D32
ICR	FDE0.0000	write only D8/D16/D32
Tx FIF0	FDE0.0200	read only D32
Rx FIF0	FDE0.0200	write only D32
	FDE0.0201	write only D32
SW-RESET	FDE0.0300	write only D8/D16/D32

Table 2: Device Addresses

For Operations on the Interface Status Register (ISR) and on the Interface Control Register (ICR), D8, D16 or D32 may be used, but only the lowest 8 bits are significant for these registers. A software reset consists in any write operation to the according address. The data transmitted by this write operation is ignored. Address FDE0.0201 (resp. FDF0.0201) is used to indicate the start of a cell on the VMEbus. The data, however, is written to the same Rx FIFO as with address FDE0.0200 (resp. FDF0.0200).

3.3 Address Modifiers

Address Modifiers for supervisor and user data access (between 30 and 3F) are responded to.

[1] transmit to VMEbus

[2] receive from VMEbus

4 ATM Interface

4.1 Physical Configuration

On the transmit side to ATM, the Fairisle VME Interface writes the cells bytewise into a FIFO (AM720035). Bit 0 of this FIFO is exclusively used to indicate the start of a cell. On its receive side from ATM, the Fairisle VME Interface expects the data to be read out of a FIFO which is used in the same way.

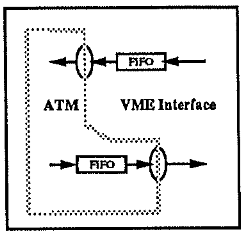

Figure 2: Interfacing to the ATM part

4.2 Cell Format

The VME Interface expects cells with a 6 byte header, a 48 byte information field . The cells have to be preceded by one control byte,whereof only bit 0 (indicating the start of the cell) is interpreted.

Byte	Name	Comment
1	PortR	Route Byte (bit 0 - set: start of cell)
2-7	Header	VCI and SAR bytes
8-55	Data	User data bytes

Table 3: Cell Buffer Format

(In this chapter the word 'byte' refers to the bits 1 to 7 of a FIFO. Bit 0 of this FIFO is set on the start of a cell.)

4.3 Cell Alignment

The Fairisle VME Interface supports an aligned transfer of the cells mentioned in chapter 4.2, i.e. the start of the information field coincides with a new D32 transfer. In this case, two VME D32 transfers are used for the header field and 12 D32 transfers for the information field. The first bit of the header field is sent on data bit 16 of the VMEbus, i.e. the lower two bytes of the VMEbus are insignificant for the first transfer. [3]

[3] General Condition for aligned transfers: (headersize mod 4 = 2) and (informationsize mod 4 = 0) preceded by one byte which indicates the start of a new cell.

5 Interface Registers

The Fairisle VME interface contains a Interface Status Register (ISR), as well as an Interface Control Register (ICR). As mentioned in chapter 3, both registers share the same device address. The Interface Status Register is read-only, whereas the Interface Control Register is write-only.

5.1 The Interface Status Register - ISR

The Interface Status Register contains data on the receive and the transmit FIFOs and on the Interrupt state. Two Interrupt Codes ic0 and ic1 indicate the reason of an interrupt.

Bit	Meaning
0	Tx FIFO logged
1	Tx FIFO half full
2	Rx FIFO logged
3	Rx FIFO half full
4	Interrupts Enabled
5	Interrupt Pending
6	Interrupt Code 0
7	Interrupt Code 1

Table 4: Interface Status Register (read-only)

5.2 The Interface Control Register - ICR

For the VME Interface being online, the front panel switch has to be set to the online position and bit 0 of the Interface Control Register has to be set. Therefore, by writing bit 0 of the Interface Control register, the host can freely take the VME Interface on or off-line. If the VME Interface is set offline, it can not receive from or transmit to the ATM Interface, [4] but all the VME operations remain supported. Bit 1 of the Interface Control Register has to be set by the host to enable any interrupts.

Bit	Meaning
0	Host Online
1	Enable Interrupts
2-7	Interrupt Vector

Table 5: Interface Control Register (write-only)

Bits 2 to 7 contain the interrupt vector base address. The value written to this location is used as the high six bits of the vector for the eight bit vectored interrupt mechanism of the VMEbus.

[4]The contents of the ATM output FIFO remain unchanged,but are not updated

6 Interrupts

6.1 Interrupt Vector

The base vector of the eight bit vectored interrupt mechanism of the VMEbus is given by the value written to the Interface Control Register.The low two bits of this interrupt vector are given by Interrupt Code 0 and Interrupt Code 1.

Base Vector	Code
7 6 5 4 3 2	1 0

Table 6: Interrupt Vector Format

6.2 Interrupt Codes

The Interrupt Codes indicate a buffer overflow in either the transmit or the receive direction. A start of cell interrupt is issued whenever there is a cell waiting for transmission on the VMEbus.

ic0	ic1	Description
0	0	Reserved
0	1	Rx FIFO full
1	0	Start of Cell Interrupt
1	1	Tx FIFO full

Table 7: Interrupt Codes

6.3 Interrupt Level

The VME interrupt level is set using links. The default interrupt level is 4. A change of the interrupt level implies two consistent changes of the link settings. Links 3-5 encode the interrupt level in binary, whereas links 6-12 have to represent the same interrupt level in unary encoding. [5] An overview of all possible link settings is given in table 1.

7 Initialisation Procedure

The following procedure is recommended in order to set up the VME Interface in a well-known state:

- Write into Interface Control Register. Set Interface Offline and enable interrupts
- Read Interrupt Status Register and verify that interrupts are enabled. (This allows to verify whether the interface board is plugged in at all.)
- Perform a software reset by a write operation to the according address.
- Write the the appropriate interrupt vector base to the Interface Control Register and set the online bit and the enable interrupt bit.

[5] Interrupt Level = (Linknumber+1) mod 7

A Slotted Ring Copy Fabric for a Multicast Fast Packet Switch

Peter Newman and Matthew Doar

The University of Cambridge Computer Laboratory
New Museums Site, Pembroke Street
Cambridge CB2 3QG
U.K.

December 1989

ABSTRACT

Asynchronous transfer mode (ATM) and fast packet switching offer the ability to support many diverse forms of telecommunications services across a single integrated network. Some applications require the communications traffic from a single source to be received concurrently by many destinations. One approach to satisfy this requirement is to use a copy fabric to replicate the necessary number of copies of the incident traffic and then to route each copy to the required destination in a conventional point-to-point ATM fast packet switch. This paper presents the design and performance of a copy fabric based upon a slotted ring that permits a very simple implementation. It offers a performance which for many applications is comparable to that of much more complex designs.

1 Introduction

Asynchronous transfer mode (ATM) has been proposed as an integrated switching mechanism for use in the public broadband ISDN and within broadband private networks. Other areas of application include metropolitan area networks (MANs) and high speed local area networks (LANs). ATM is a transfer mode that uses fast packet switching with short fixed length packets to reduce the delay and the variance of delay for delay sensitive services such as voice and video [?]. These short packets are called cells and currently broadband ISDN proposes a cell length of 48 octets of information with 5 octets of header [?]. Many ATM fast packet switch designs only support unicast operation, i.e. point-to-point operation, where one incoming cell produces one outgoing cell routed to a single specific destination. There are a number of applications that require the switches of the network to

[1]To appear in Proc. Int. Switching Symposium, Stockholm, Sweden, May 1990.

be capable of multicast operation where a single incoming cell produces multiple outgoing cells. Each outgoing copy of a multicast cell contains the same information but is routed to a different destination. Such applications include video conferencing, audio conferencing, entertainment video distribution, the interconnection of local area networks and some aspects of distributed processing.

There are a number of possible approaches to the design of a multicast ATM fast packet switch but many of the existing designs lead to a very complex implementation. The approach proposed in this paper yields a very simple implementation that may easily be implemented in current gate array technology. It may be prefixed to any existing ATM switch design to offer multicast operation without requiring any modifications to the switch. The design was developed to serve both the private broadband networks market and within the high speed local area backbone network where switch designs that require vast areas of custom silicon are at present inappropriate.

2 Design of a Multicast Fast Packet Switch

The simplest means to achieve multicast operation is for the source to send multiple copies of each multicast cell to each destination, one after the other. This simple technique has several major drawbacks. The delay between the generation of the first and last copies of each multicast cell will be large. The source must waste valuable time individually transmitting many copies of the same cell. Also, network resources would be more efficiently utilised if copies of multicast cells were not generated until later nodes in the network wherever possible. A slightly more efficient approach uses a dedicated server at each switching node to make the required number of copies of every incident multicast cell and to route each copy individually to its required destination [?]. The copying and routing is still performed in series, however, thus a large delay may still exist between the generation of the first copy and the last. This approach is therefore not suitable for the multicast operation of real-time traffic such as voice or video and may not be fast enough for some distributed computing applications.

For such applications the operations of copying and routing must be performed in parallel, i.e. all of the copies of each incident multicast cell must be copied and routed concurrently. Also, many different multicast cells from separate switch ports should be capable of being handled at the same time. For ATM fast packet switch designs that use a single path switch fabric the operations of copying and routing the multicast cells may be implemented in the same switch fabric that handles the switching of the unicast traffic, but such switches may not be constructed with a total switch capacity above a few Gbits/sec in current technology [?, ?]. ATM fast packet switch designs that use a multi-stage switch fabric with output buffered switching elements may also implement multicast operation within the same switch fabric that performs the routing function but this makes the design of the switching element very complex [?, ?]. Also, non-buffered multi-stage switch fabrics exist that

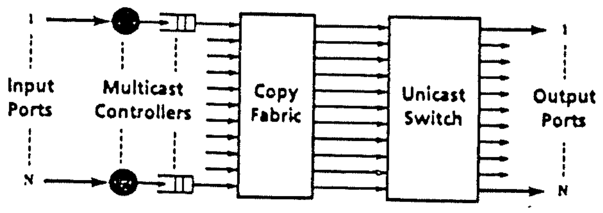

Figure 1: General structure of a multicast ATM fast packet switch.

can handle both unicast and multicast traffic within a single switch fabric, e.g. the Richards network [?] and also Lea [?]. In general, however, these tend to be more suited to synchronous transfer mode (STM) with centralised control than for the distributed control that is essential for a high capacity ATM fast packet switch.

A more general solution is shown in fig. 1. A unicast (point-to-point) ATM fast packet switch is preceded by a copy fabric and a set of multicast controllers. The multicast controllers add a copy tag to each of the incoming cells and store them in an input buffer. The copy tag defines the number of copies of the cell that are required. The copy fabric then generates the required number of copies of each cell in parallel and each copy exits from the copy fabric on a separate output port at approximately the same time. All cells that exit from the copy fabric, whether originally unicast or multicast, may now be handled in the same manner by the unicast switch and routed to their respective destinations.

3 Banyan Copy Fabric Designs

Two designs of copy fabric have been proposed in the literature based upon a banyan interconnection network constructed from 2×2 broadcast switching elements (nodes). Each broadcast switching element has the ability to route an incident cell to either one of its two outputs, or to generate two copies of the cell, one on each output. Turner's switch [?, ?] uses buffered nodes in the copy fabric. A fanout field, appended to the front of each cell, specifies the required number of copies and each node in the interconnection network uses this field to determine whether to produce a single copy of each incident cell or two copies. The copy fabric is blocking in that incident cells may contend for the same resources (nodes and links) within the copy fabric therefore each node must have at least a single cell buffer on each input. Busyback signals are used between the nodes in each stage of the network to indicate a full cell buffer. Due to the buffering, each copy of the cell is not guaranteed to exit the copy fabric at the same time and there is some randomness built into the network as to the outputs of the network from which each copy will emerge.

Lee's copy fabric [?] is also based upon a banyan network but is non-blocking and also non-buffered. The fanout field at the head of each incident multicast cell is presented to a running adder network which computes a range of contiguous output ports from which the required number of copies will emerge. This address range is prefixed to the cell as a minimum and a maximum output address. All cells are applied to the copy fabric in alignment and from the minimum and maximum address fields each node may decide on which output to route the cell or whether to produce two copies. In this copy fabric all copies will exit the fabric at exactly the same time but the ports upon which they will exit are dependent upon the other traffic within the network. Furthermore, for Lee's copy fabric to work, active ports may not be interspersed with inactive ports, thus a concentration fabric is required prior to the copy fabric.

The above examples of copy fabrics do not offer simple implementation in current gate array technology. They are also difficult to partition into a single component that may be replicated in order to implement copy fabrics of any size. There are many applications that do not require the extremes of performance that a banyan copy fabric may be able to offer. Also, many applications cannot justify the investment in custom silicon required to implement such copy fabrics. The following is a very simple design of copy fabric that may easily be implemented in current gate array technology yet it offers a multicast capacity that for many applications may not be greatly inferior to the above examples.

4 A Slotted Ring Copy Fabric

4.1 Design

The proposed copy fabric is based upon a slotted ring and is illustrated in fig. 2. Each node on the ring supports a single input port and a single output port of the copy fabric. A multicast controller is connected to each input port of the copy fabric and the output ports of the copy fabric feed into the unicast switch as illustrated in fig. 1. Sufficient storage (typically one or two octets) is located in each node of the ring for an integer number of slots to circulate on the ring. Each slot is long enough to contain a complete cell. The slot structure is illustrated in fig. 3 and consists of the copy tag followed by the cell. The copy tag specifies the required number of copies of the cell. The slot structure may be maintained by a single timing source and distributed to each node of the ring as the copy fabric is likely to be implemented within a single cabinet.

ATM fast packet switching requires that all cells have a header that contains a label. The label defines which connection each cell belongs to. Each multicast controller performs a table look-up on the label of every incident cell and extracts from the table the copy tag associated with the connection to which that cell belongs. It prefixes the copy tag to the front of each incoming cell and stores the cells in the input buffer which is a first in first out (FIFO) queue.

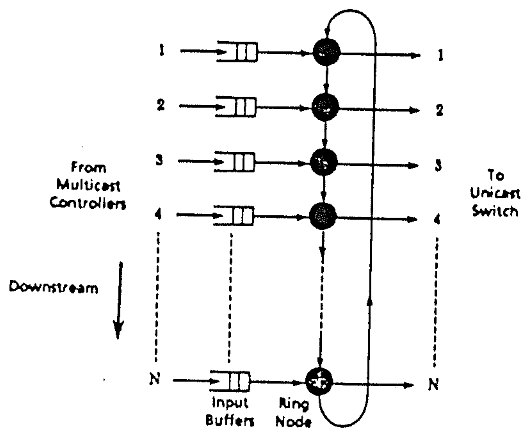

Figure 2: The slotted ring copy fabric.

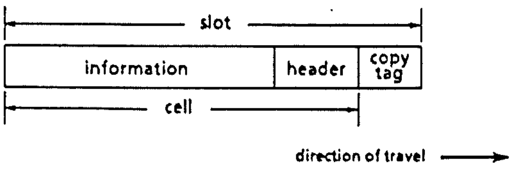

Figure 3: Slot structure.

When a node on the copy fabric has a cell waiting in its input buffer it waits for the beginning of the next slot to arrive on the ring. If that slot is free the cell is written into that slot else if the slot is full the node waits for the next free slot. A slot is considered free if the copy tag at the head of the slot is zero. As a full slot passes each node on the ring a copy of the cell is transmitted to the output port of that node and the copy tag at the head of the slot is decremented by one. Thus the required number of copies of the cell will exit from the copy fabric, the first copy from the output port of the node at which the cell entered and further copies from consecutive downstream ports. As soon as the required number of copies of the cell have been produced the copy tag becomes zero hence the slot is free for use by the next node on the ring that has a cell waiting. The copy tag may need to be removed from each cell on exit from the copy fabric to restore the cell to the format expected by the unicast switch.

Many proposals exist for the use of a slotted ring as a switching fabric, e.g. [?, ?]. In a slotted ring used as a switching fabric each cell must exit on a specific output port of the ring depending upon its required destination. For traffic with a random distribution of destinations each cell will on average have to travel half way round the ring to reach the destination it requires. Thus, on average, for a single slot ring two cells may be serviced for each complete rotation of the slot.

The use of a slotted ring as a copy fabric differs from its use as a switching fabric because the copy fabric places no restrictions upon which output ports of the copy fabric the cells must exit. The only constraint is that each copy of a cell exits the copy fabric on a separate output port. Each copy of a cell may therefore exit the ring on the nearest available output port. This occurs when each copy of a cell exits the ring on consecutive downstream output ports. Thus under saturation, during a single rotation of a single slot ring, every output port of the ring may produce a copy of a cell. This compares with a maximum of two ports producing output cells for a slotted ring used as a switching fabric. This will remain true whatever the distribution of copy requests in the copy tags. Thus a slotted ring used as a copy fabric in the above manner has a far greater capacity than a slotted ring used as a switching fabric.

Fig. 4 illustrates a single slot ring copy fabric under saturation for one complete rotation of the slot. Although every input is assumed to have a multicast cell waiting to access the ring only five have gained access to the ring during a single rotation of the slot. Every output, however, is busy producing one copy of a cell for every rotation of the slot therefore the copy fabric is operating at saturation. (The illustration of fig. 4 suggests that all cells exit the copy fabric with their headers aligned, whereas in reality there will be a small delay of a few bit times between the start of each cell due to the storage in each node of the ring.)

4.2 Label Allocation

In a unicast ATM fast packet switch the connection to which a cell belongs is identified by the label, or virtual channel identifier (VCI), in the cell header. In general each input port of a unicast switch allocates its own labels. A multicast cell will be replicated in the copy fabric and identical copies of the cell with identical labels will appear at a number of input ports to the unicast switch. Care must therefore be taken to ensure that there is no conflict between the labels of unicast and multicast traffic. A simple method of doing so is to allocate unicast labels from the bottom end of the address space and multicast labels from the top of the address space. A more efficient method is to translate the labels of multicast cells as they pass through the multicast controllers. Thus each port may select both its unicast and multicast labels independently of all other ports and the translation of multicast labels in the multicast controllers permits any conflict between labels to

Figure 4: An example of copy fabric operation at saturation.

be avoided. All copies of multicast cells emerge from the copy fabric on fixed output ports assigned at call setup. Thus the translated multicast labels need not be unique across all input ports of the unicast switch but only across those input ports that belong to the same multicast connection. So the same translated multicast label may be used for any number of multicast connections that do not have any input ports of the unicast switch in common.

In this simple slotted ring copy fabric the output port of the copy fabric at which each copy of a multicast cell will exit is fixed and known at the time the connection is established. This removes the requirement for label translation at the output of the copy fabric and permits the copy fabric to be used with any design of unicast switch without requiring that switch to be modified. In more complex schemes, e.g. Lee [?], any copy of a multicast cell may emerge at any output port of the copy fabric depending upon the current traffic load within the copy fabric. In this case every output port of the copy fabric requires a translation table that contains a label translation for every copy of each of the multicast labels. This rapidly leads to a very large translation table on every output of the copy fabric even for moderate sizes of switch.

4.3 Enhancements

Unfortunately, at high loads the copy fabric may become unfair in that some busy input ports may 'hog' the ring preventing downstream ports from gaining a fair share of access to the ring. If the bandwidth of the ring can be made sufficiently high this effect may not become important, otherwise a mechanism that ensures fairness may need to be implemented. One possible mechanism is to use a counter-rotating ring containing a single bit per ring node which is used to indicate which downstream nodes have cells to send. Upstream nodes may therefore take into account the traffic of downstream nodes before inserting their own cells onto the ring in a similar manner to that of the DQDB metropolitan area network proposal [?]. This mechanism might also be used to include priority information into the ring access algorithm to ensure that high priority traffic is served before traffic of lower priority. This is likely to be important for delay sensitive multicast traffic such as conference voice and video in the presence of delay insensitive traffic [?].

It is possible that in some applications most multicast cells will only require a few copies to be made. In this case, if the mean multicast traffic load is fairly low, a ring much smaller than the number of input ports may be used as a copy fabric. Each node of the copy fabric would handle the multicast traffic from several input ports and any multicast cells that require more copies than the size of the copy fabric could circulate around the ring more than once. The information in the copy tag attached to each cell would have to be used to distinguish between multiple copies of the same cell arriving at the same output port of the copy fabric.

In the above discussion it has been assumed that unicast cells (which do not require multiple copies) are handled in the same manner as multicast cells. The copy tag for a unicast cell is set to one and it passes through the copy fabric in exactly the same manner as multicast traffic. It emerges on the output port of the node at which it entered without further copies being made. It therefore suffers the same delay through the copy fabric as multicast traffic. If the delay performance for unicast traffic is to be optimised it would be better if unicast traffic were routed directly to the unicast switch without passing through the copy fabric. Fig. 5 illustrates one method of achieving this. A larger unicast switch is used and the outputs of the copy fabric feed into separate input ports of the unicast switch. The copy fabric need not be the same size as the number of input ports to the multicast controllers as several multicast controllers may share the same input port of the copy fabric. This design has the advantage that all multicast traffic is handled on separate ports of the unicast switch from the unicast traffic. This prevents the multicast traffic of one port from interfering with the unicast traffic of a downstream port which is a disadvantage of the former design. A concentrating interconnection network may be used as the switch fabric of the unicast switch if a large number of multicast ports are required.

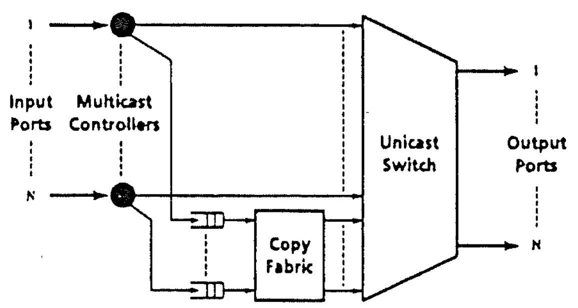

Figure 5: Switch structure for enhanced performance.

The performance of the copy fabric at high loads will be impaired if the incident traffic is not evenly distributed across all inputs of the copy fabric. For a high performance copy fabric an even distribution of traffic across the copy fabric may be ensured by placing a distribution fabric before the copy fabric. Cells arriving at any of the input ports of the switch may be switched through the distribution fabric to any of the input ports of the copy fabric. All cells belonging to the same connection must follow the same path through the the distribution and copy fabrics to avoid out of sequence errors between cells on the same connection. The distribution of the traffic must therefore be performed on a per call basis at call setup and may be based upon an estimate of the current load on each section of the ring. One possibility for the implementation of the distribution fabric is to use another slotted ring since the load on the distribution fabric will be lower than that on the copy fabric.

5 Performance

5.1 The Simulation Model

The performance of the copy fabric was investigated using a simulation model. Copy fabrics of size 16 ports to 256 ports were modelled. It was assumed that each node on the ring possessed the required number of bits of storage in order that an integer number of slots fitted exactly onto the ring. The results are presented for the case in which exactly one slot fits onto the ring as the performance does not vary with the number of slots on the ring. The slot is assumed to be exactly one cell in length and all delay results are normalised to the length of a cell. A cell is assumed to be available for transmission as soon as it begins to arrive, i.e. the copy fabric operates in cut-through mode. Delays are measured from the time a cell begins to arrive until it commences transmission across the copy fabric. Each input port is assumed to carry slotted traffic and the incident load on any port gives the probability of any slot containing a valid cell. All input ports are assumed to be equally loaded and the phase relationship between the slot structure arriving at each input port is assumed to be random.

Each multicast cell will request multiple copies to be generated from a minimum of two to a maximum equal to the size of the copy fabric. The number of copies requested is called the fanout. Two fanout distributions were modelled: fixed and geometric. In the fixed fanout model all fanouts are the same fixed value. In the geometric case the fanout of each multicast cell is given by a truncated geometric distribution with parameter p similar to that used in [?]. The probability that the fanout is k is given by

$$\text{Pr (fanout} = k) = \begin{cases} p(1-p)^{k-2} & 2 \leq k < N \\ (1-p)^{N-2} & k = N \end{cases}$$

where $0 \leq p \leq 1$ and N is the size of the copy fabric. The mean fanout obtained from this distribution is given by

$$E \text{ (fanout)} = \begin{cases} (1/p)[1-(1-p)^{N-1}]+1 & 0 < p \leq 1 \\ N & p = 0 \end{cases}$$

As each incident multicast cell gives rise to multiple outgoing cells the applied load is given by the product of the incident load and the mean fanout and is normalised to the throughput per port at saturation of the copy fabric. Each simulation run was allowed to reach stability and then measurements were taken for a total of 200,000 cells. This yielded measurements of mean delay with a standard deviation of about 6% of the mean for low loads and about 3% of the mean for medium to high loads.

5.2 Delay

The mean delay of a copy fabric of size 64 ports is given in fig. 6 for a geometric fanout distribution with mean fanouts from 2 to 32. The 99th percentile of the delay distribution under the same conditions is given in fig. 7. The delay is at a maximum for a mean fanout of about half of the size of the ring (a fanout of 32 in this case) and beyond this fanout the mean delay decreases.

For a fixed fanout distribution the mean delay for fanouts up to 8 does not differ significantly from the geometric fanout distribution. For higher mean fanouts the fixed distribution gives a mean delay of up to about 30% greater at loads above 0.5 but a reduced 99th percentile of delay.

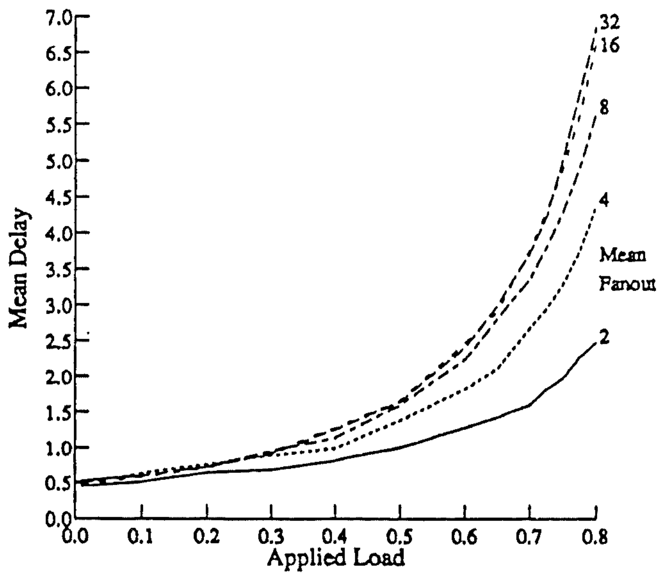

Figure 6: Mean delay for a 64 port copy fabric.

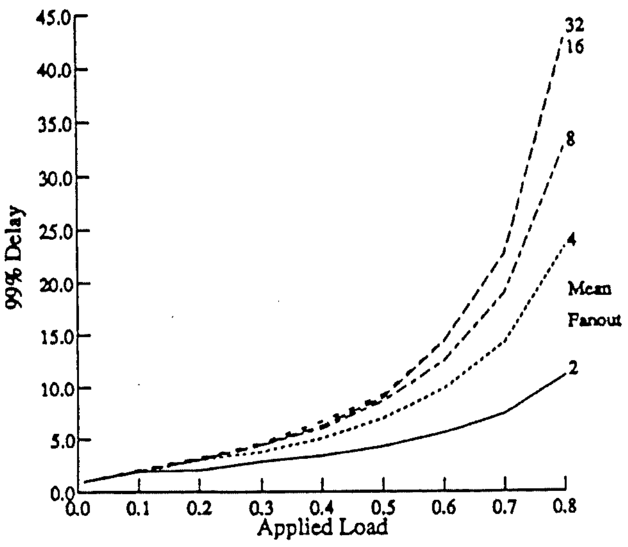

Figure 7: 99th Percentile of delay distribution for a 64 port copy fabric.

It may be seen that for an applied load of 0.7 the mean delay is about 4 cell lengths and the 99^{th} percentile of delay about 20 cell lengths. This is comparable with the mean delay performance for the buffered banyan copy fabric reported in [?]. It also corresponds very closely to the delay performance of a unicast input buffered ATM fast packet switch operating at 80% of its throughput at saturation [?, ?]. Thus assuming a maximum load of 0.7 the copy fabric gives a very similar delay performance to a unicast input buffered switch operating at its maximum load if the incident traffic is randomly distributed across all inputs.

5.3 Size of Copy Fabric

Fig. 8 shows that the above measurements of mean delay do not vary greatly with the size of the copy fabric for small values of mean fanout. Fig. 9 illustrates how the mean delay varies with the mean fanout for various sizes of copy fabric and a geometric fanout distribution at an applied load of 0.7. It may be seen that the mean delay reaches a maximum when the mean fanout is about 30% to 50% of the size of the copy fabric.

5.4 Mixed Traffic

The simplest implementation occurs when unicast traffic passes through the copy fabric together with the multicast traffic. In this case the unicast traffic will experience the same delay as the multicast traffic. Fig. 10 shows how the delay through the copy fabric is reduced as the ratio of unicast traffic to multicast traffic is increased for a copy fabric of size 64 with a mean fanout of 8. The probability of any incident cell being a unicast cell is given by the unicast traffic ratio. Unicast traffic has a fanout of unity whereas multicast traffic has a fanout given by the geometric distri-

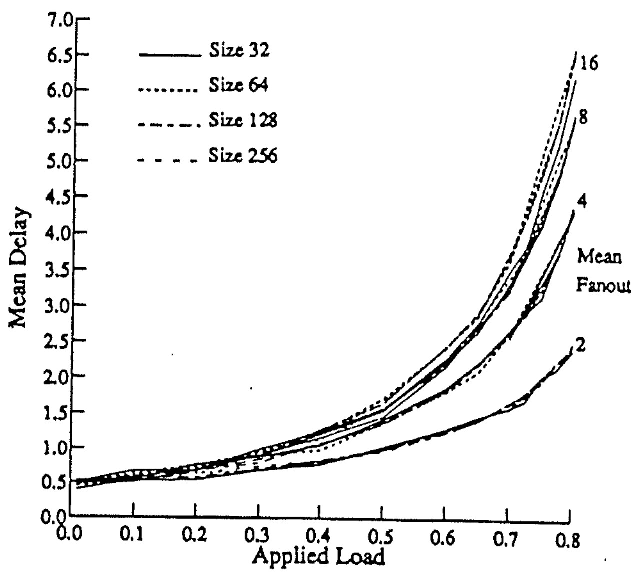

Figure 8: Mean delay for various sizes of copy fabric.

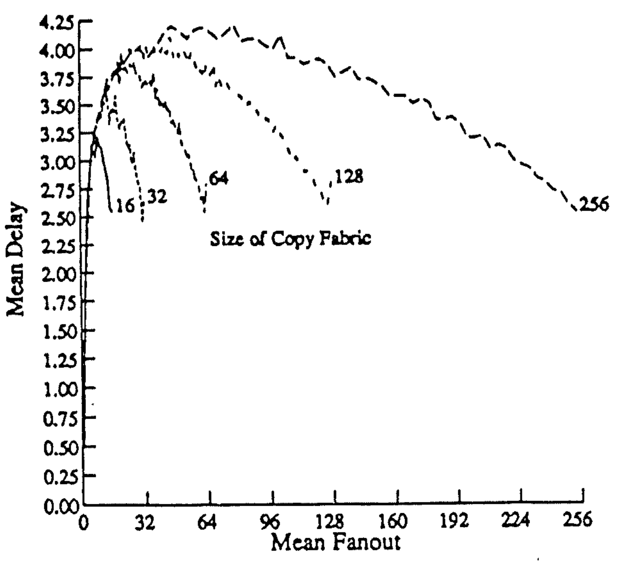

Figure 9: Variation of mean delay with mean fanout for various sizes of copy fabric at an applied load of 0.7.

bution thus the effective mean fanout of the traffic mix depends upon the unicast traffic ratio. The applied load is adjusted to take into account the effective mean fanout of the traffic mix.

For a unicast traffic ratio of zero all traffic is multicast and the curve is the same as that of fig. 6. For a unicast ratio of one all traffic is unicast and the delay is 0.5 which represents the average time any cell must wait for the arrival of the head of the slot on the ring given a random phase difference between traffic arrivals on each input port. It may be seen that the presence of a relatively small amount of multicast traffic in the traffic mix rapidly increases the mean delay to approach that of pure multicast traffic.

5.5 Fairness

The basic design of the copy fabric exhibits sensitivity to the distribution of the incident traffic across the input ports. For example, if two consecutive input ports are heavily loaded the downstream port will receive much poorer service than the upstream port. A simple solution to the problem is to make the bandwidth of the ring much greater than that of the input ports. This is not difficult to implement as the ring may be constructed up to 8 or 16 bits wide and clocked at up to 50 MHz in current CMOS gate array technology. This would offer a basic throughput at saturation approaching 1 Gbit/sec on every port.

An alternative approach is to implement a fairness algorithm. A counter-rotating ring with a single bit per input port may be used to inform upstream ports when a cell is waiting in an input port further downstream. Each input port may use this information in deciding whether to transmit a cell or to defer to a downstream port with a cell waiting. Investigations have shown that while not offering perfect

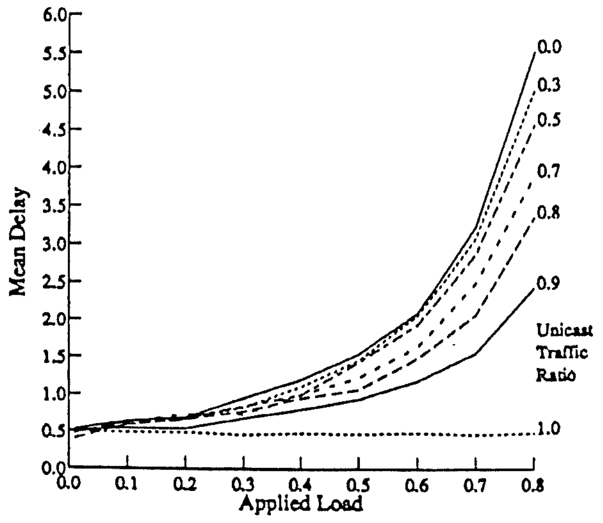

Figure 10: Variation of mean delay with traffic mix for a 64 port copy fabric and a mean fanout of 8.

fairness this approach can ensure that all ports receive a reasonable share of the bandwidth even under the worst case conditions.

6 Implementation

The slotted ring copy fabric is currently being implemented as part of an enhanced design of the Cambridge Fast Packet Switch [?, ?]. The objective is to produce a low cost ATM fast packet switch that can be used in the experimental investigation of ATM networking. From this work a design will be produced that is relevant to private wide area networks and also as a high speed local area backbone network.

In order to reduce the experimental hardware required the copy tag will be inserted as a field in the cell header. This permits the function of the multicast controller to be replaced by a simple FIFO provided that the label translation function in the unicast switch also translates the copy tag. The copy fabric has been designed using an 8 bit wide ring clocked at 25 MHz which gives a throughput at saturation of 200 Mbits/sec for each port. Thus, in this implementation, a 64 port copy fabric operating at a maximum load of 0.7 will offer a total traffic capacity in the region of 10 Gbits/sec. Each node of the copy fabric may be constructed in less than 500 gates using current CMOS gate array technology. This allows the multicast function to be integrated into the input port controllers of the unicast switch.

7 Conclusions

A copy fabric based upon a slotted ring has been proposed that may be prefixed to any unicast ATM fast packet switch to support multicast operation. The performance of the copy fabric has been investigated and shown to be comparable with much more complex designs provided that the incident traffic is evenly distributed across the input ports. The even distribution of traffic across the copy fabric may be ensured by the use of a distribution fabric before the copy fabric but in general this should only be necessary for switches designed for high multicast loads. For a copy fabric operating at a maximum load of 0.7 the mean delay is about 4 cell times and the 99^{th} percentile of delay about 20 cell times. This translates to a mean delay of about 12 μsecs and a maximum delay for 99% of the traffic of about 60 μsecs for standard ATM cells at 150 Mbits/sec. This is very similar to the performance that would be expected from an input buffered unicast ATM switch operating at its maximum load of 80% of its throughput at saturation. The performance does not vary with the number of slots on the ring and varies only slightly with the size of the ring.

The slotted ring copy fabric offers a very simple implementation in current gate array technology. This makes it most appropriate for integration with a simple design of unicast ATM switch such as the Cambridge Fast Packet Switch [?, ?]. In this case it may be implemented within the input port controllers of the switch prior to the label translation function. The resulting switch design is most appropriate to the emerging ATM private networks market and also within high speed local and metropolitan area networks.

Fairisle Connector

Notes:

"In" and "out" are with respect to the fabric, (ie fabric is in, transmission line is out).

There are essential two types of interface sync and async; sync uses frame sync and phi, async uses Fifo control.

The B row is VME (J2) compatible. (Ribbon connectors will not pickup row B.)

Pins a1-a3 and a30-32 are either ground/power for port controller to fabric connection or multicast signals (m0-m5) in case of port controller daughterboard connector.

	a	b	c
1	GND (m0)	5V	GND
2	GND (m1)	GND	GND
3	GND (m2)	VME	GND
4	dOut[0]	VME	GND
5	dOut[1]	VME	GND
6	dOut[2]	VME	GND
7	dOut[3]	VME	GND
8	dOut[4]	VME	GND
9	dOut[5]	VME	GND
10	dOut[6]	VME	GND
11	dOut[7]	VME	GND
12	NC / SOCOut	GND	GND
13	ackForOut / OutFifoRBar	5V	GND
14	fsForOut / OutFifoEBar	VME	GND
15	phi2 / NC	VME	GND
16	phi1 / NC	VME	GND
17	dIn[0]	VME	GND
18	dIn[1]	VME	GND
19	dIn[2]	VME	GND
20	dIn[3]	VME	GND
21	dIn[4]	VME	GND
22	dIn[5]	GND	GND
23	dIn[6]	VME	GND
24	dIn[7]	VME	GND
25	NC / SOCIn	VME	GND
26	ackForIn / InFifoRBar	VME	GND
27	fsForIn / InFifoEBar	VME	GND
28	NC	VME	GND
29	NC	VME	GND
30	5V (m3)	VME	5V
31	5V (m4)	GND	5V
32	5V (m5)	5V	5V