The Desk Area Network

by D. R. McAuley and I. M. Leslie

The authors describe their experiences with a novel architecture, the 'Desk Area Network', for use within a computing end-system. The architecture extended several concepts used in modern high-speed networks into computer system design, with the goal of building a platform able to supply 'quality of service' in support of multimedia applications.

1 Introduction

The paper presents an overview of the work on the 'Desk Area Network' (DAN), a computer system architecture to address the problems raised in supplying high-bandwidth channels between the components of a modern desktop computer when there are constraints on the required bandwidth and timeliness of the individual channels. This problem arises in the context of supplying end-to-end 'quality of service' (QoS) to support networked real-time multimedia applications.

The first Section presents the QoS problem in the networking context and then focuses on the problem as faced by end-systems attached to an ATM (asynchronous transfer mode) network. The solution described is based on the use of an ATM *switch fabric* for processor, memory and device interconnection inside computer systems in place of more traditional bus architectures.

The software architecture resulting from this work had a key role in shaping the Nemesis operating system[1] and has been shown to have qualitative benefits even on more traditional bus-based platforms.

2 Background

A goal of one thread of recent and ongoing research in networking, computer architectures and operating systems is to engineer systems which can support 'quality of service'; that is the desire to provide systems where access to shared resources is provided in terms of lower bounds on the amount of resource and upper bounds on timeliness of access.

As an example, consider a simple video monitoring application in which video is digitised and compressed by a device attached to the network, shipped across the network into a computer, where it is both displayed in real time and processed to perform motion detection, and then, if motion is detected, recorded to hard disc. Clearly the demand on the disc is variable and indeed, if the compression is based on a variable-bit-rate algorithm (e.g.

Abbreviations

AAL	=	ATM adaptation layer	HEC	=	header error check
ADC	=	analogue-to-digital converter	I/O	=	input/output
ATM	=	asynchronous transfer mode	JPEG	=	Joint Photographic Experts Group
CPU	=	central processing unit	LAN	=	local-area network
CSMA/CD	=	carrier sense multiple access with collision detection	MIPS	=	million instructions per second
			NTSC	=	National Television System Committee
DAN	=	Desk Area Network			
DAT	=	digital audio tape	PAL	=	phase alternate line
DMA	=	direct memory access	PCI	=	peripheral component interconnect
DRAM	=	dynamic random access memory	PDU	=	protocol data unit
DSP	=	digital signal processing	QoS	=	quality of service
EPROM	=	electrically programmable read only memory	RISC	=	reduced instruction set computer
			SRAM	=	static random access memory
FDDI	=	fibre distributed data interface	TDM	=	time division multiplex
FIFO	=	first-in first-out	VCI	=	virtual channel indentifier
FPC	=	Fairisle port controller	VHS	=	Video Home System

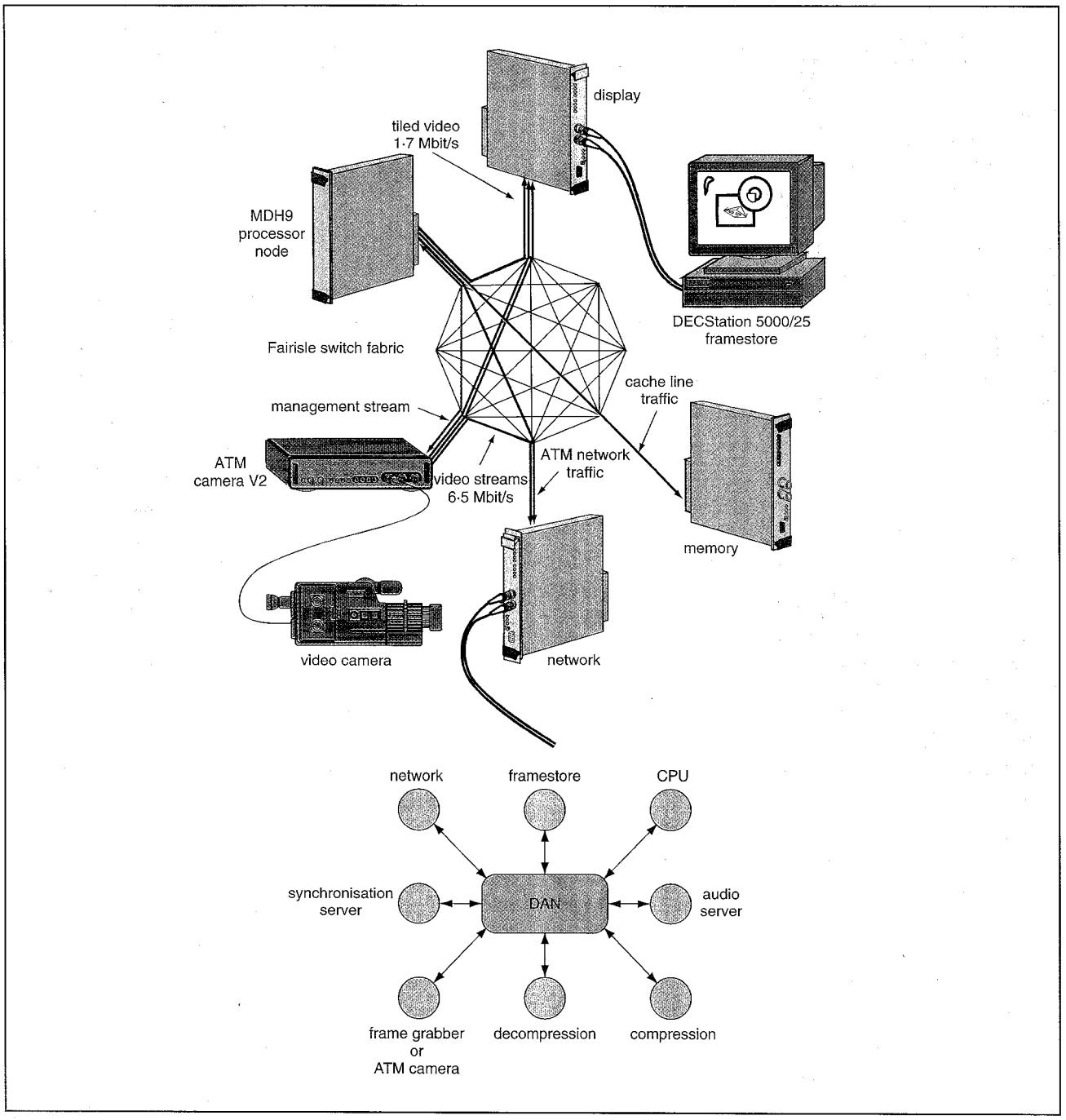


Fig. 1 The DAN concept

to achieve constant quality), the demands placed on the network, computer bus and processor are likewise variable.

There are two key questions to be asked at this point:

(*a*) Is the system to be built solely for this purpose or do we wish to use more-general-purpose network and computer systems in which there are other tasks competing for resources?

(*b*) Must guarantees be provided *all of the time* or just *most of the time*?

Increasingly the answer to the first question is that a generic infrastructure is desired. Sharing and reusing resources is often more cost effective, while it is desirable that the system be sufficiently flexible to adapt to new services, protocols and applications as they arise.

If we require that resource be available *all of the time*, that is the application requires cast-iron guarantees, then we must perform 'peak-rate allocation', that is reserve an amount of the resource equal to the peak that the application *might* demand. In a shared environment this means it is possible to admit new applications as long as the sum of the peak demands is less than the amount of resource.

For our example video monitoring application, consider the network element. Assume some bandwidth properties of the video encoder and define the capacity of some link shared by several such streams of data — say the encoder has a peak rate of 10 Mbit/s and a mean rate of 1 Mbit/s,

and that the shared link has a capacity of 20 Mbit/s. If we insist on cast-iron guarantees we could only admit two such streams onto the link.

On the other hand, if we allow the possibility of 'overbooking' the bandwidth, we might admit many more channels and hence increase average utilisation at the risk of delay or loss; in networking parlance this is 'statistical multiplexing'. Traditionally data networks (e.g. X.25, Ethernet, Internet) have engaged in statistical multiplexing without any attempt to provide a quantified (non-zero!) minimum service level for individual channels; indeed the main goal of the design work in media-access-control protocols like token passing and the CSMA/CD algorithm of Ethernet is to aim for a fair share of the bandwidth resource to all active participants.

However, the desire to move real-time traffic (e.g. video and audio) through networks will often require some minimum guarantee: consumers will demand that video-on-demand is at least something approaching 'VHS quality'; likewise audiophiles listening to concerts over the net will demand something approaching the quality they are used to from radio broadcasts; there may be legal issues in future — digital security video may have minimum requirements to be admissible in court.

So the 'quality-of-service' problem is to provide a channel with constraints on the delay, jitter (i.e. variation in delay) and loss characteristics in the face of a variable demand about which we may have only the simplest statistics, such as peak and mean rates, while at the same time trying to maximise the utilisation.

Both the Internet and ATM communities are working hard to provide network QoS; indeed much of the theoretical work on characterising traffic, defining effective scheduling and queueing algorithms in switches, and call acceptance is applicable in both domains.

The discussion so far has concentrated on network QoS to illustrate the issues; however, from the application's point of view, achieving a suitable QoS in the network is only part of the problem — we must achieve end-to-end QoS, which implies QoS within the computer itself. That is we must ensure the computer system is ready and able to receive the data, ensures timely dispatch of the application code, and then supplies sufficient processor and disc capacity for the application to do its job; again this must be achieved in the face of simultaneous demands from other tasks.

3 The Desk Area Network

The DAN machine (Fig. 1) was built to explore the provision of QoS in a computer interconnect. We selected an ATM-cell-based switch fabric as the primary processor, device and memory interconnect. A particular goal was to interconnect continuous media devices (e.g. audio and video devices) attached to different DAN systems over an ATM local-area network (LAN), in order to provide the underlying channels for distributed real-time multimedia applications.

One element of the work focused on the design of devices capable of acting as sources or sinks for streams of media data injected into or received from the ATM network directly — data should only be handled by the host

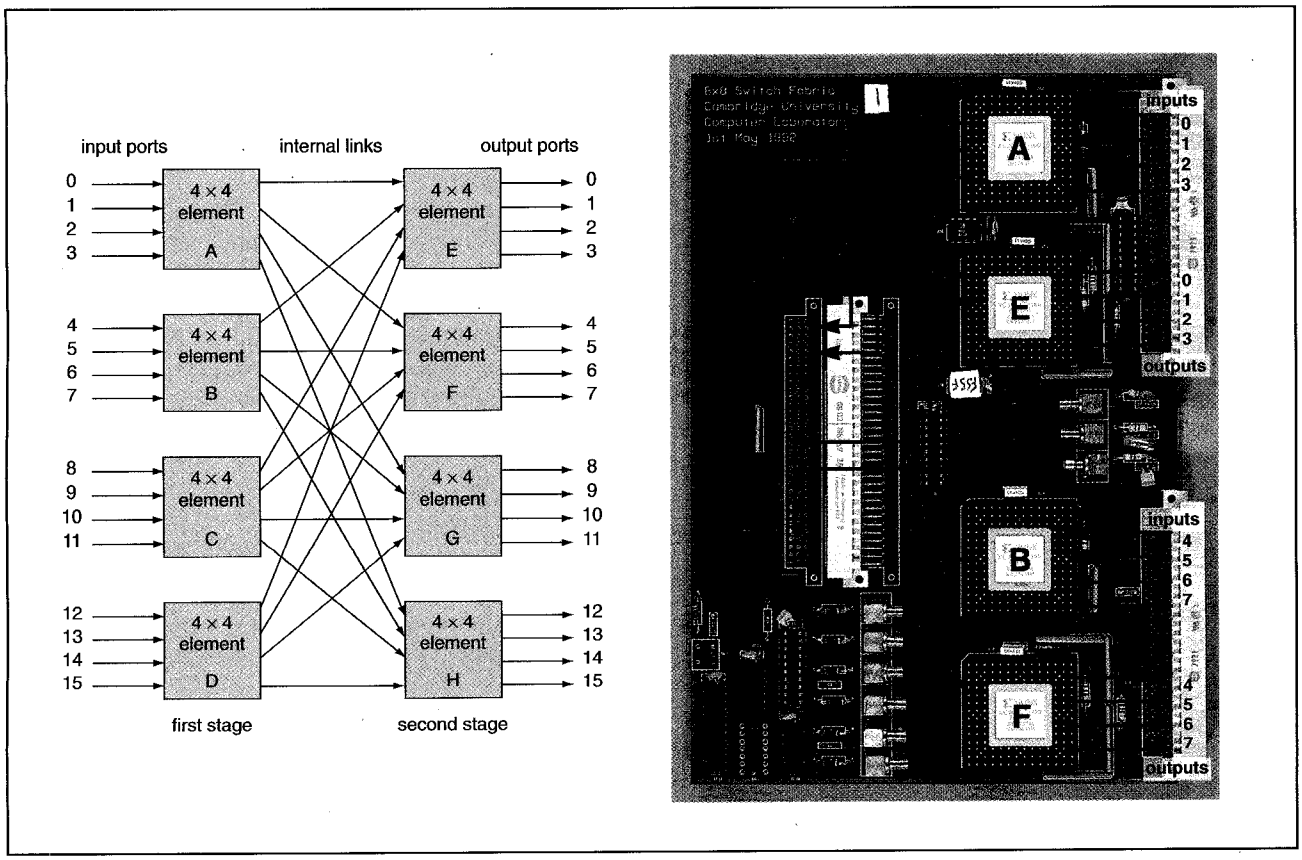


Fig. 2 The Fairisle switch fabric

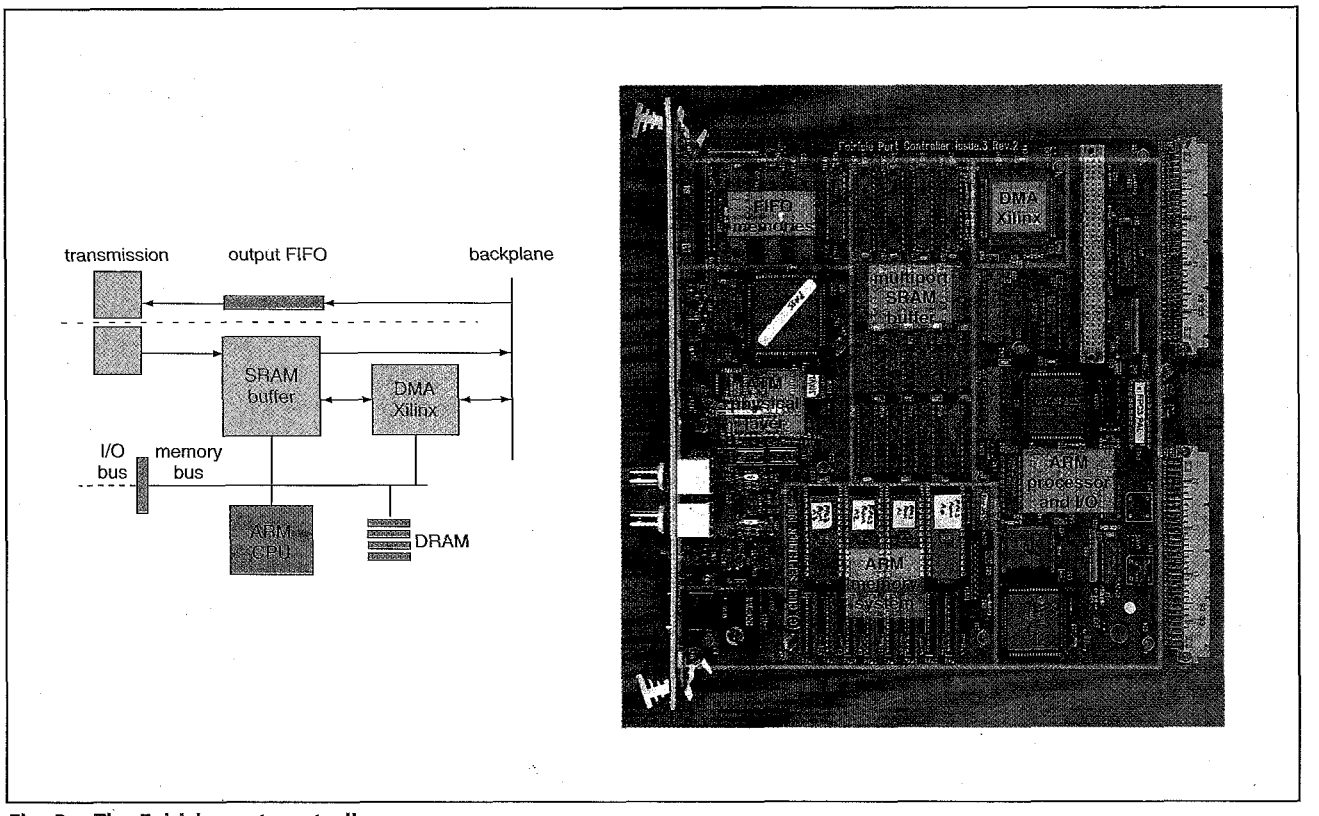


Fig. 3 The Fairisle port controller

processor when it requires processing. A further underlying design criterion was to minimise synchronous real-time control operations on the device as much as possible; doing this reduces the real-time scheduling demands on the processor and allows pipelining of operations for greater throughput.

Another element of the work addressed the operating system and control issues raised by the desire to enable applications running on the host processor to control the communications between the various devices. The processor is in effect performing third-party channel set-up and is not normally involved in the actual data flow. How then does the application running on the processor monitor the progress of the data flow?

The DAN fabric

The experimental DAN system was built using a space division switch fabric designed and built by the Fairisle project.[2] This provided a 16 port switch fabric with each port 8 bits wide and clocked at 20 MHz; although this is a fairly modest bandwidth per port, the 2·56 Gbit/s maximum system bandwidth is quite respectable. The fabric layout is shown in Fig. 2.

One relevant property of the Fairisle fabric is that of *blocking*: should two or more cells contend for the same output, or an internal link within the fabric, one will succeed in getting through and the others will remain at the relevant input until the next cycle.

All communications between attached devices are based around the exchange of sequences of cells. Some view the chopping up of arbitrarily sized messages between devices into fixed-size chunks with some distaste. However, such techniques are already widely used in currently fashionable busses such as SUN SBus and PCI, and for the same reason, namely to enforce re-arbitration of the system interconnect at frequent intervals so that urgent small messages can pre-empt long transfers. In current workstations and PCs this facility is widely used to ensure that a processor cache line fetch or flush can pre-empt long disc and network DMA (direct memory access) transfers.

Given the desire to choose some small-sized transfer into which to fragment messages we selected to use the now standardised ATM cell format;[3] in common with many ATM switch designs, we use 48 bytes of payload and 4 bytes of header internally, the 1 byte header error check (HEC) being consumed and generated by the transmission system. As our view was that the primary network to which the DAN system would attach was an ATM LAN this choice made the network interface card extremely simple.

A key issue in ensuring the provision of QoS within the DAN fabric is the ability to control the rate at which devices inject cells into the fabric on different channels (and hence to different output ports); the devices described below can all be rate-limited by controlling software running one of the processor nodes of the DAN

Although we chose to use our particular fabric, the work on the devices discussed below is in fact independent of the exact realisation of the ATM fabric. For example, any time division multiplex (TDM) system, whether bus, dual bus, folded bus or ring, could just as easily be used if a rate-controlled access is implemented in the arbitration.

The DAN devices

Four devices were produced for the DAN machine:

- video capture (the so called 'ATM camera')

- video display (the 'DAN framestore')
- audio input/output and DSP (digital signal processing)
- processor and cache node.

Three functional units were implemented by modification of the Fairisle port controller:

- main memory node
- ATM interface
- Ethernet interface

We present the key features in outline here, further details are available in Reference 4.

The Fairisle port controller: The use of Fairisle ATM switch fabric as the DAN fabric has already been mentioned; the Fairisle port controller (FPC — see Fig. 3) was the second element of the Fairisle switch which we were able to reuse for the DAN machine by modification of the embedded software.

In the Fairisle switch, for each bidirectional link* attached to the switch there is an FPC card. The card includes the transmission system realising the ATM link, and buffer memory arranged as a number of queues. A first-in-first-out (FIFO) queue at the output suffices, as its role is simply to deal with the mismatch in clock frequencies between the output of the switch fabric and the transmission line. However, the blocking properties of the Fairisle fabric dictate that when the fabric is heavily loaded queues will build up at the input; as part of the Fairisle work was to investigate various queueing strategies, a reduced-instruction-set (RISC) processor was used to manage the queues — in the final FPC v.3 card, an ARM600 processor.

Given this functionality it then becomes clear that an unmodified FPC can be used as the 'network interface' for the DAN machine; it was simply required to do the same

* We use 100 Mbit/s 4B5B-coded links — in fact the FDDI physical layer.

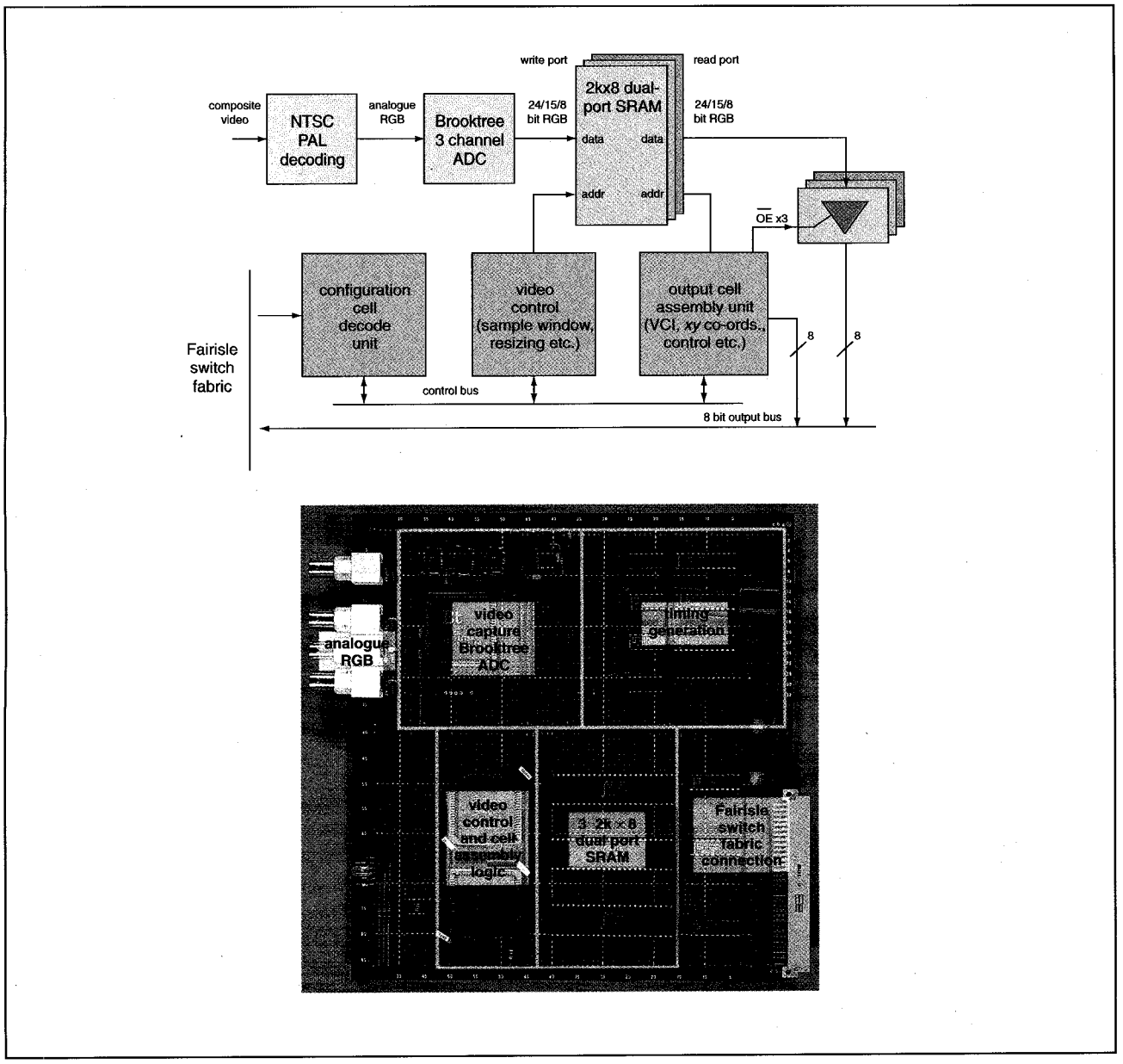


Fig. 4 The prototype ATM camera

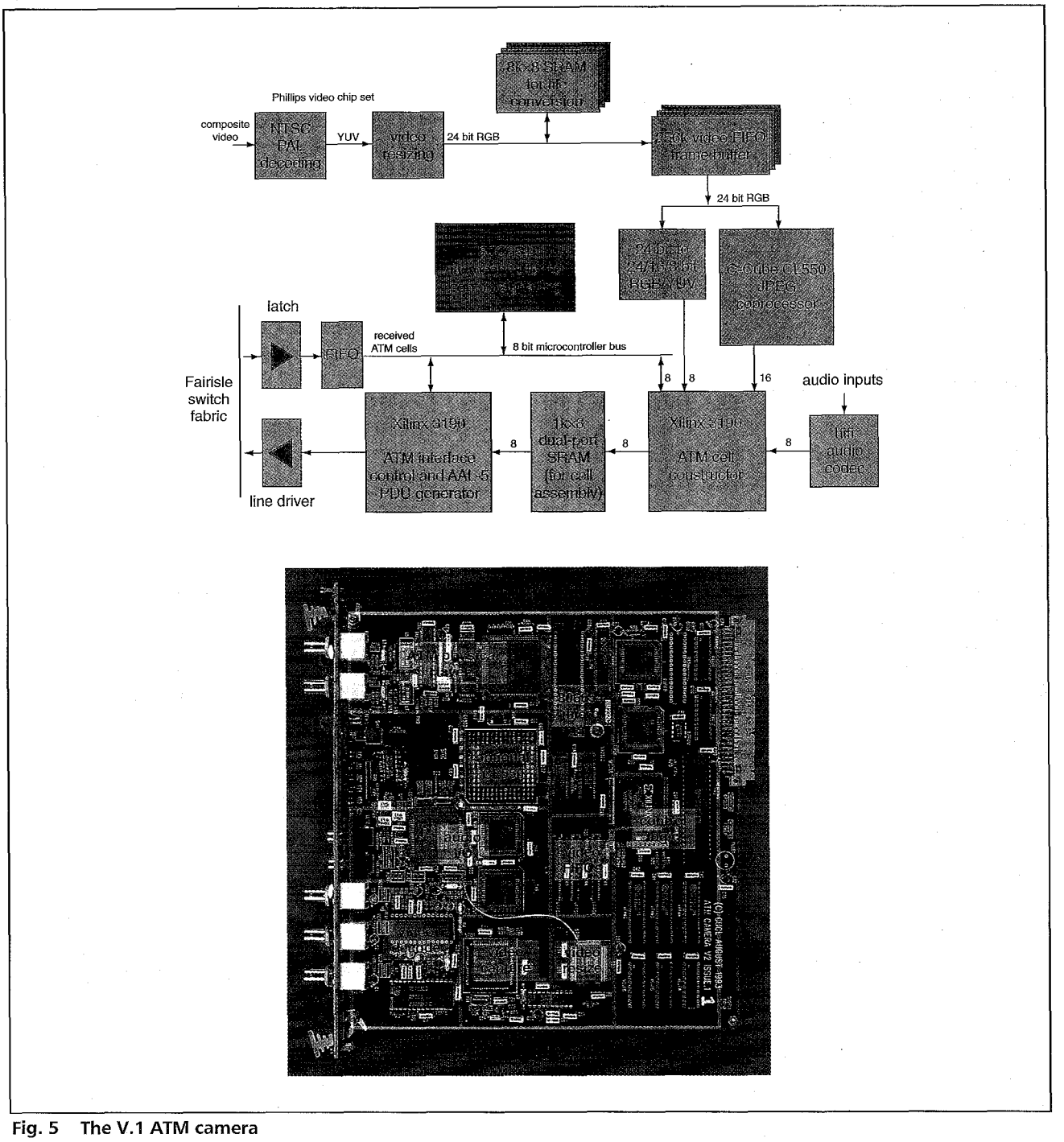


Fig. 5 The V.1 ATM camera

job as in a switch, namely to transfer cells between the LAN link and the internal DAN fabric.

Finally, each FPC contains an input/output (I/O) connector enabling an Ethernet interface to be fitted; then, with a modification to the software configuration, the ATM interface could also be used as an Ethernet interface.

ATM camera: To gain experience with the DAN, the first hardware built was a prototype ATM camera (Fig. 4). Although this was capable of direct device-to-device transfers it required synchronous updating of various registers on the card to enable it to be multiplexed between multiple video sources or to change format between video fields. The latter was seen as critical as one important use of the ATM camera in our scenario was to generate several resolutions of the same video stream and to select one to send to a particular participant, based on their display capabilities or network access. However, with the prototype we demonstrated the simplicity of working with ATM cells as the basic unit of transfer.

The V.1 was the final research version of the ATM camera produced.* The V.1 (Fig. 5) has the ability to generate digital video streams from up to three sources simultaneously (if they are genlocked) in a number of formats (24 bit RGB, 15 bit RGB, 8 bit RGB, 8 bit monochrome, 16 bit YUV) as sequences of 8×8 pixel tiles packed within an ATM adaptation layer AAL5 frame.

* Nemesys Research Ltd. has since added considerably more functionality in the move to commercial production.

Optionally, JPEG compression can be applied to the tiles inside the AAL5 frame. To provide resilience to cell (and hence packet) loss, we chose to compress the data in each AAL5 frame independently. This ensures that if an AAL5 unit carrying a set of tiles is lost, only the tiles in that AAL5 unit are affected. This technique is conformant with standard JPEG coding; any JPEG device should be able to decompress and display the image. We simply use reset markers to delineate the frames, trading a small increase in bandwidth against greater resilience to cell loss. Raw and compressed video can be produced simultaneously subject to the 160 Mbit/s limit associated with the ATM fabric to which the camera is attached.

A microcontroller on the board is used to select source, resolution, format, compression option and destination virtual channel identifier (VCI) per video field. This selection is based on a cyclic schedule loaded by the camera manager software in response to requests from applications — the schedule contains 50 (or 60) entries, one per field of PAL (or NTSC) video. The main processor need only interact with the device whenever the schedule needs to be changed. Fig. 5 shows the increased complexity of the device over the original prototype, partly to increase the functionality (e.g. to add JPEG compression etc.) and partly to add some dedicated processing, in the form of the microcontroller, to the card. However, the complexity is still on a par with bus-based cards of similar functionality.

Framestore: Early work with the DAN used off-the-shelf components attached to the switch fabric and software to emulate the framestore; in particular a DECStation 5000/25 workstation (25 MHz MIPS R2000 processor[5]) was used. This allowed simple testing of the ATM camera and experiments with various versions of the tiled video format.

The goal of investigating the required functionality of a DAN framestore device was performed using this emulated set-up. At one extreme of functionality, the framestore ran a complete X-server with video extensions and a process which provided an emulation of a Pandora box;[6] at the other, the framestore contained only interrupt routines which rendered 8×8 pixel tiles.

Later, a framestore was purpose-built for the DAN (see Fig. 6). The high-capacity path into the framestore allowed multiple streams to be displayed simultaneously. Control and data paths are conceptually separate. Control operations allow a window manager to create, destroy, move and resize windows and for clients to be connected to windows. For data transfer the framestore provides only a function to render a pixel tile at an offset within a particular window, each window being defined by an offset from the origin and bounding box. A nearly fully functional X-server was developed to use the framestore over the DAN.

The framestore incorporates novel protection features which prevent incoming streams from writing to pixels belonging to another window. This is now in the process of being patented.

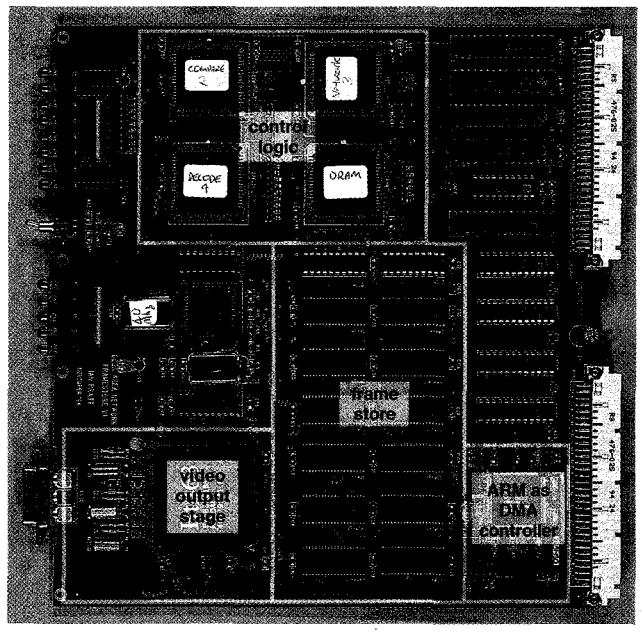


Fig. 6 The DAN framestore

Audio input/output: A general-purpose digital signal processing and audio codec node was built using Analog Devices' ADSP-2111 DSP chip and associated 'DAT quality' audio codecs. The memory system for this processor was constructed to optimise operations on stream data which were destined to be received from the ATM switch fabric and, after processing, retransmitted to another DAN node.

This board (see Fig. 7) illustrates that the interface to the ATM switch fabric need not be particularly complex; indeed, in terms of gate count, the DAN interface is considerably simpler than many popular busses.

DAN processor node: A processing node was developed which attaches directly to the switch fabric, through which it communicates with its main DRAM-based memory. The so-called MDH9 (Fig. 8) is composed of a general-purpose RISC processor with first-level cache on chip (ARM600) and a large external second-level cache. The second-level cache is a *write back* cache — on a cache miss, an ATM cell containing the data and address of a cache line to be flushed is sent over the DAN to the DRAM memory system; this cell also contains the read address for the new cache line to be fetched so that the memory system can return the cell with the required data.

Today, four years after construction, the memory access time (5 μs) and bandwidth (160 Mbit/s) clearly show their age, although at the time these measures were not out of line with mid-range workstations. However, the major reason for pursuing the switch-based attachment for cache/memory was to investigate the impact of direct insertion of high-bandwidth media streams into the processor cache for applications requiring access to the media data. Results from investigations using the MDH9 are published in Reference 7; these demonstrated the need for a more flexible cache controller, the design and construction of which awaits a new project.

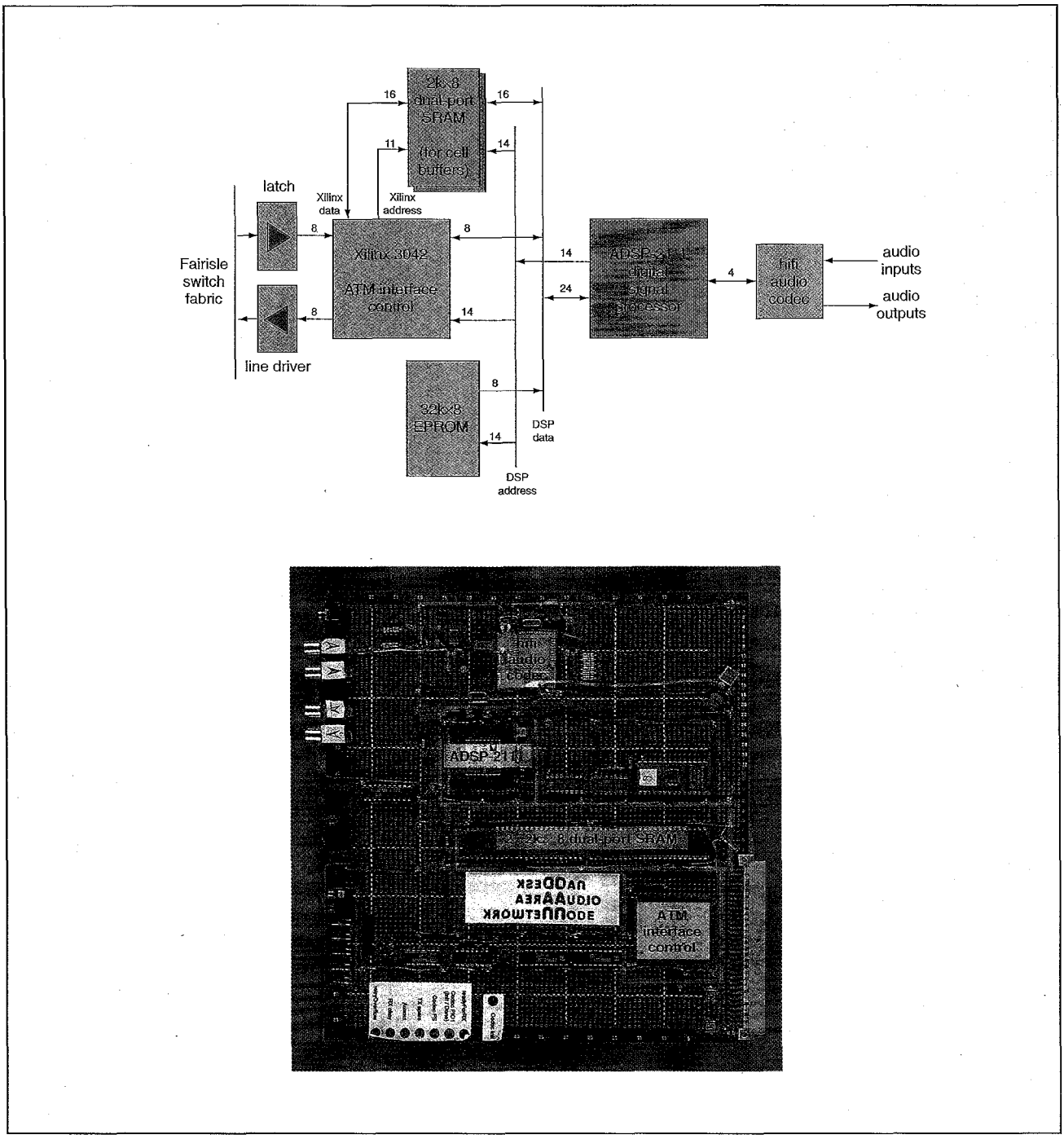


Fig. 7 The DAN audio node

Protocols

The DAN mechanism of device attachment differs from approaches based on 'networked devices' in that the components are attached by means of a reliable and secure communication channel to a processing node that is responsible for their management. In the same manner as a bus-based system, this allows the control protocol to be comparatively simple — is is not necessary to implement protocol modules to deal with communication errors and encryption modules for security. Devices present a heterogeneous set of control interfaces as appropriate to their function — the control of a device can be as simple as a register read/write interface as in the prototype ATM camera, or more involved, such as in the case of the DSP node, where control is achieved by downloading appropriate code.

The separation of control and data paths is a key to the DAN architecture. Plugging streams from one device into another, with a processing element being involved only in the set-up phase, relies on this separation. Indeed it can be argued that many of the interesting features of the DAN are to do with its connection-oriented nature rather than the use of ATM *per se*.

Another use of the separation of control and data is the provision of synchronisation information. A node, e.g. a general-purpose processor, synchronising two video streams needs only to look at the time stamp information derived from the video, not at the video streams

themselves. For this reason we have defined a common interface and protocol for synchronisation events; these form a low-rate cell stream (e.g. one cell per video field) to the appropriate manager. These individual synchronisation streams are then available to a synchronisation service within the DAN for onward propagation or local adaptive control as appropriate.

Unless it is desired to process the media data, the main processor of the system only deals with the synchronisation and control messages, while the data transfer operations are performed independently and directly between the attached devices (including the network interface device). This requires that the data stream generated by the device contains all the in-band information required by the receiver (e.g. multiplexing and sequencing information). Requiring the stream to pass through the processor to have this information added would defeat one of the main objectives of the DAN approach.

As an example, we can consider the encoding of video: video is transmitted in (AAL5) frames composed of a number of 8×8 pixel tiles, with each frame containing in-band control information, such as the X/Y offset of the tile within the frame, and the frame number; however, to simplify the in-band protocol, advantage is taken of channel set-up and renegotiation to communicate parameters which change on a long time-scale — for example, frame size, frame rate, and compression options.

This general approach has been presented in Reference 8.

Software

The first runtime system used within the ARM-processor-based nodes of the DAN was the Wanda micro-kernel, a locally developed operating system used by a number of projects. This provided a familiar environment in which to implement the device management functions. Indeed both the software emulations for the experimental frame buffers (the X-server with Pandora video extensions together with the Pandora box emulator, and the very much simpler video rendering code) were implemented over Wanda.

Later, in conjunction with the ESPRIT Pegasus project,[9] a version of the Nemesis operating system was ported to the port controller processors. Nemesis is concerned with providing QoS guarantees to applications; in some respects it is the operating-system analogue of the DAN. Nemesis follows the philosophy that the operating system should not be involved in the data transfer between two applications, or between an application and a device. The operating system should limit itself to the control functions, for example in establishing connections between an application and a device.

The novel features of the DAN provided a good environment for Nemesis and many features in Nemesis can be traced to the requirement that it run on the DAN. In particular the DAN is a multiprocessor without shared memory; this meant that the natural synchronisation paradigm was event counts and sequencers,[10] hence event counts and sequencers became the native synchronisation

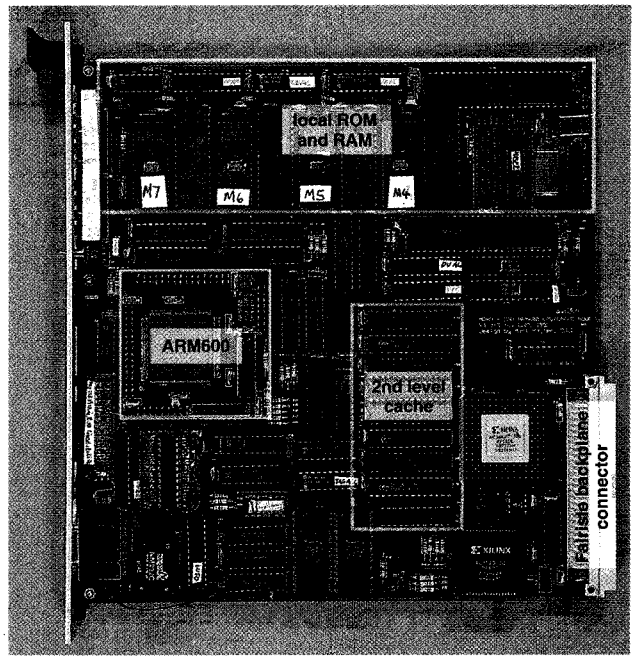


Fig. 8 The MDH9

paradigm for Nemesis, even when run on uniprocessor workstations. The benefit for uniprocessors becomes apparent with the new processor designs using superscalar techniques and out-of-order memory accesses; the need to insert *memory barrier* instructions, which enforce the ordered access but reduce processor performance, is greatly reduced. Other synchronisation primitives (semaphores, POSIX threads) were implemented over event counts and sequencers.

The I/O architecture of the DAN has also influenced the I/O architecture of Nemesis, again even when run on uniprocessor workstations. The DAN enforced consideration of how to separate the control and data paths of I/O transactions while maintaining efficiency. The same view when applied to a uniprocessor machine has strained some of the underlying hardware but has also indicated how little functionality can be required to increase performance significantly. For example, the frame buffer driver on the DEC Alpha workstation implementation of Nemesis emulates the DAN framestore. This enables client applications of the framestore to write directly to the frame buffer device (using the same pixel tile primitive as on the DAN framestore), bypassing the operating system and window system, *but still maintaining protection* — one application cannot draw on another application's pixels.

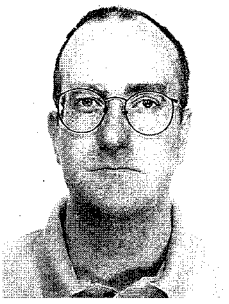
The mainstream Nemesis work has also demonstrated the same approach applied to the provision of QoS in disc access and the network interface. This current state of this work is presented in Reference 11.

4 Conclusions

Our DAN-based approach to a multimedia workstation can be contrasted with two approaches seen in the industry today: either throw more processing power at the problem or supply dedicated paths for audio and video.

Ian Leslie received a BASc degree in 1977 and an MASc degree in 1979 both from the University of Toronto and a PhD from the University of Cambridge Computer Laboratory in 1983. Since then he has been a university lecturer at the Cambridge Computer Laboratory. His current research interests are in ATM network performance, network control, and operating systems which support guarantees to applications.

Address: Computer Laboratory, University of Cambridge, New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK



Derek McAuley received a BA degree in Mathematics in 1982 and a PhD in Computing Science in 1988 from the University of Cambridge. After five years as a lecturer at the Computer Laboratory in Cambridge, he moved to a chair in the Department of Computing Science at Glasgow University. His current research includes machine architectures and operating systems support for time-constrained media applications.

Address: Department of Computing Science, University of Glasgow, 8–17 Lilybank Gardens, Glasgow G12 8RZ, UK

The first is in one sense simple to implement and indeed provides the greatest functionality — move everything via the memory system and get the processor to do the tedious task of routing the data between devices. However, this usually suffers badly from artefacts of the I/O subsystem due to the simplistic schemes by which the I/O capacity is allocated — it is easy to obtain unacceptable performance for video and audio if there are other tasks performing I/O, even when the system is lightly loaded. All of this can happen even when the system is running an operating system alleging to provide QoS.

The second approach simplifies the hardware and guarantees the data gets where it is going on time by supplying dedicated hardware resource. However, if such data paths restrict the allowable formats and the programmability is limited, the cost is the reduced generality, and hence difficulty in adding new applications.

The DAN work has taken a middle ground and demonstrated key results:

(*a*) It is possible to build an I/O system within a computer which is both general purpose and delivers QoS to real-time media streams.
(*b*) With careful consideration of the control and data path separation, such an I/O system can be realised with hardware of similar complexity to traditional busses.

The DAN used ATM as a way to ensure the fine-grained sharing of bandwidth within the system while simplifying the attachment of the system to an ATM LAN. The latter was viewed as important in our scenarios, where a significant proportion of the load on the interconnect was between the network interface and the other devices on the DAN.

Whether one would choose to use an ATM-style interconnect may be down to a matter of taste; however, the key lessons we have learnt are:

(*a*) implement the bandwidth-sharing to ensure the requisite real-time granularity is achieved for the media streams
(*b*) implement a scheduling mechanism for the interconnect that provides software-controlled sharing of the proportions allocated to the individual channels
(*c*) ensure the behaviour of the scheduling scheme for some channels is defined and meets their real-time requirements even when the applied load is in excess of capacity
(*d*) separate control and data paths cleanly so that device-to-device data transfer can be established and controlled by a third party.

References

1 LESLIE, I., McAULEY, D., BLACK, R., ROSCOE, T., BARHAM, P., EVERS, D., FAIRBAIRNS, R., and HYDEN, E.: 'The design and implementation of an operating system to support distributed multimedia applications', *IEEE J. Sel. Areas Commun.*, September 1996, **14**, (7), pp. 1280–1297
2 BLACK, R. J., LESLIE, I. M., and McAULEY, D. R.: 'Experiences of building an ATM switch for the local area'. Proceedings 1994 SIGCOMM Conf., London, UK, August–September 1994
3 The ATM Forum: 'ATM user-network interface specification. Version 3.0' (Prentice Hall, 1993)
4 BARHAM, P., HAYTER, M., McAULEY, D., and PRATT, I.: 'Devices on the Desk Area Network', *IEEE J. Sel. Areas Commun.*, May 1995, **13**, (4), pp. 722–732
5 Digital Equipment Corporation Workstation Systems Engineering: 'MAXine system module functional specification. Revision 1.2', February 1991
6 HOPPER, A.: 'Pandora — an experimental system for multimedia applications', ACM Operating Systems Review, 1990
7 HAYTER, M.: 'A workstation architecture to support multimedia'. Technical Report 319, University of Cambridge Computer Laboratory, September 1993. PhD dissertation
8 McAULEY, D.: 'Operating system support for the desk area network'. Proceedings NOSSDAV'93, Lancaster, UK, November 1993
9 MULLENDER, S. J., LESLIE, I. M., and McAULEY, D. R.: 'Operating-system support for distributed multimedia'. Proceedings Summer 1994 Usenix Conference, Boston, Massachusetts, USA, June 1994, pp. 209–220. Also available as Pegasus Paper 94–6
10 REED, D., and KANODIA, R.: 'Synchronisation with event counts and sequencers'. Technical report, MIT Laboratory for Computer Science, 1997
11 BARHAM, P.: 'Devices in a multi-service operating system'. PhD thesis, University of Cambridge Computer Laboratory, 1995 (in preparation)

First received 30th September 1996 and in revised form 19th February 1997