Design and Implementation of a Versatile Multimedia Network Interface and I/O Chip[†]

Zubin D. Dittia
zubin@dworkin.wustl.edu

Jerome R. Cox, Jr.
jrc@hobbs.wustl.edu

Guru M. Parulkar
guru@flora.wustl.edu

Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130.

Abstract

We present an I/O architecture that is particularly well suited to the transport and delivery of continuous media data to devices distributed in the desk or home area. Our design makes use of the APIC chip, which is a high performance (1.2 Gb/s) research prototype of an ATM network interface that we have developed at Washington University. Both hardware and software issues are considered. Due to space constraints, this extended abstract will only touch on aspects of our architecture that we consider important. More details will be presented in the full-length version of the paper.

[†] This work was supported in part by the Advanced Research Projects Agency (ARPA), the National Science Foundation (NSF), and an industrial consortium of Ascom Nexion, Bay Networks, Bell Northern Research, NEC America, NIH, NTT, Samsung, Southwestern Bell, and Tektronix.

Extended Abstract

Introduction

Although gigabit network design has seen major advances in the last decade, it is a well known fact that delivering gigabit bandwidths to end applications is not easy. The bottlenecks that prevent applications from exploiting all or most of the available network bandwidth manifest themselves in both the hardware and software realms. From a hardware perspective, limitations have traditionally been imposed by the peak system memory and bus bandwidths. By rethinking the host I/O subsystem, and moving processing power closer to network I/O, it is possible to bypass the main system memory and bus altogether. This approach is especially well-suited for continuous media data (e.g., video, audio, etc.), where the processing is usually of a very specialized nature (e.g., signal processing, compression, encryption, etc.) and needs to be carried out in real-time. From a software viewpoint, operations such as data copying, software checksumming, frequency of servicing interrupts, and context switching are responsible for poor performance. By carefully designing the network interface, OS, and protocols in a coherent and integrated fashion, it is possible to minimize, or in some cases to completely annul, the handicaps resulting from the above operations.

The APIC (*ATM Port Interconnect Controller*) chip is a high performance 1.2 Gb/s ATM network interface chip developed by us at Washington University. It differs from traditional network interfaces in the fact that it has a small 2×2 ATM switch built into the chip, which allows multiple APICs to be interconnected to build desk area networks (DANs). An APIC desk-area network can be thought of as a high performance cell switched replacement for an I/O bus. The APIC is rich in the mechanisms it provides to improve performance. In particular, it includes ways to reduce or eliminate data copying, provide assistance to the OS with transport checksumming, reduce the frequency of interrupts, etc. A detailed overview of the APIC design was presented in [3].

In this paper, we describe how media devices interface to APICs, and how these devices can be remotely controlled. We will see that the remote controllability is very important if we are to be able to build home-area multimedia networks, which connect up various multimedia and other devices in the home to a central control computer (the legendary set-top box).

For the rest of this paper, we will make the tacit assumption that ATM can serve as a universal data communication substrate (along the lines of RS232) for all I/O devices used in offices and homes. The rationale behind this assumption will be presented in the full length version of the paper.

DAN Architecture

A desk area network (DAN) is an architecture in which a packet switch serves as a workstation interconnect. Work on desk area networks was pioneered at the University of Cambridge [6,8], and was independently pursued at MIT by Tennenhouse [7].

A high level schematic of our proposed DAN architecture is shown in Figure 1. At the heart of this architecture is a daisy-chained interconnect comprising of a number of APIC chips. Individual chips interface either directly to the main system bus, or through a dual ported memory (e.g., VRAM) to an I/O device One of the APIC chips on the interconnect is connected to the external ATM network through a link interface. ATM cells arriving from the network pass from one APIC to another until they reach an APIC chip that is directly connected to the destination device (or to the system bus). Similarly, data originating from a source device is passed as an ATM cell stream from the APIC directly connected to that device, down the APIC interconnect towards the optical link interface, and finally out to the network. It is also possible for two local devices to communicate over the APIC interconnect, without cells ever having to leave the desk area. Thus, for example, a video cell stream originating at a camera could traverse the interconnect in order to reach the local display device. The APIC which directly interfaces to the system bus can be used for traffic originating from, or destined for, the system's main memory. All other traffic movement happens

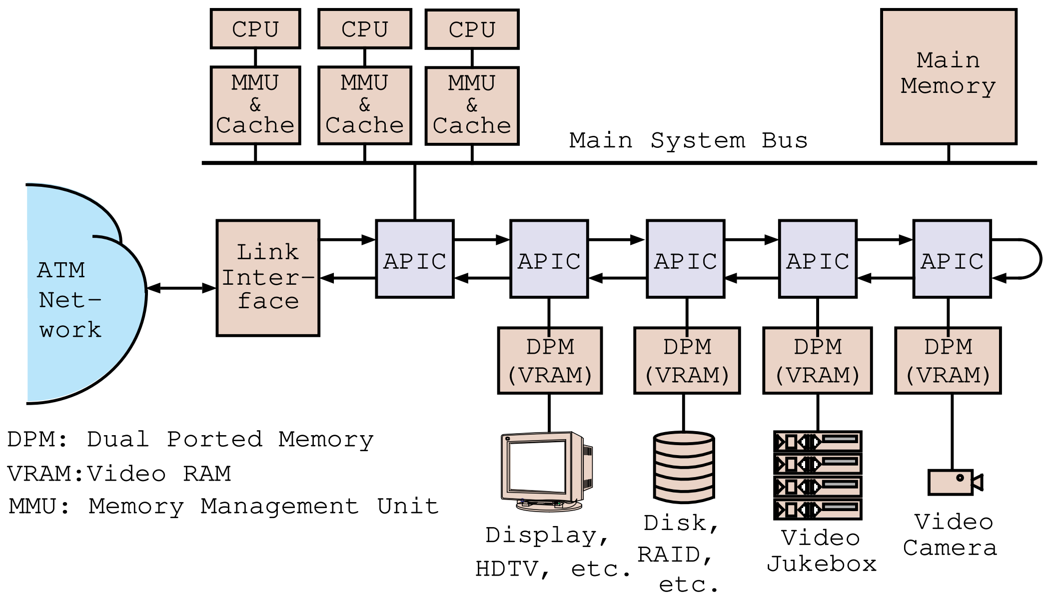

Figure 1: Desk Area Network Architecture.

only over the APIC interconnect, thus sparing the main system bus of this burden.

In our prototype implementation, each APIC will interface to a neighboring APIC over two 1.2 Gb/s links. ATM Forum's UTOPIA (Universal Test and Operations PHY interface for ATM) [1] standard is used on these links. The data path width is 16 bits. The prototype APIC supports two buses on its external memory interface: Sun's MBus, which is a 2.56 Gb/s peak (64-bit multiplexed address/data at 40 MHz), and PCI [5], which is 1.06 Gb/s peak (32-bit multiplexed address/data at 33 MHz).

Since the APIC can be thought of as a 2×2 ATM switch with an on-chip network interface, other topologies for the DAN are also possible. For instance, the two ATM ports can be used to interconnect APICs in a perfect shuffle, manhattan street network, or delta network topologies.

The APIC chip, which is central to our design, is responsible for taking specified data from the memory, creating an ATM cell stream using that data, and forwarding the cell stream to a downstream APIC. In the reverse direction, the APIC is responsible for doing a VPI/VCI virtual channel lookup on every arriving cell to determine if it is destined for the local device. If so, the cell payload is placed in memory, else it is forwarded to a downstream APIC. The APIC is also responsible for implementing full AAL-5 segmentation and reassembly; thus, frames up to 64 Kbytes in length can be reassembled. Entire frames are not reassembled within the APIC itself; instead, the APIC gradually builds a frame in the external memory by writing the payload from a few cells at a time. The APIC is also responsible for *pacing* outgoing cells at a pacing rate that is independently specifiable on a per-connection basis. This serves two purposes: first, it ensures that the APIC's local device does not get an unfair share of the ATM interconnect bandwidth, and second, it prevents bursty sources from causing network congestion or from swamping the receiver with a burst that cannot be handled (by the receiver). In addition to the above-mentioned features, the APIC chip also supports multipoint connections, mechanisms to reduce the frequency and overhead of servicing interrupts, and DMA mechanisms that are designed to allow the OS to implement zero-copy protocol stacks.

Media Processing

The architecture of the DAN as we have described it, provides no easy way to do any processing on data that remains restricted to the APIC interconnect and does not visit the system bus. For continuous media data, it is important to be able to do media-specific processing such as compression, encryption, etc. We have two approaches for tackling this problem. The first approach is to have a special media processor

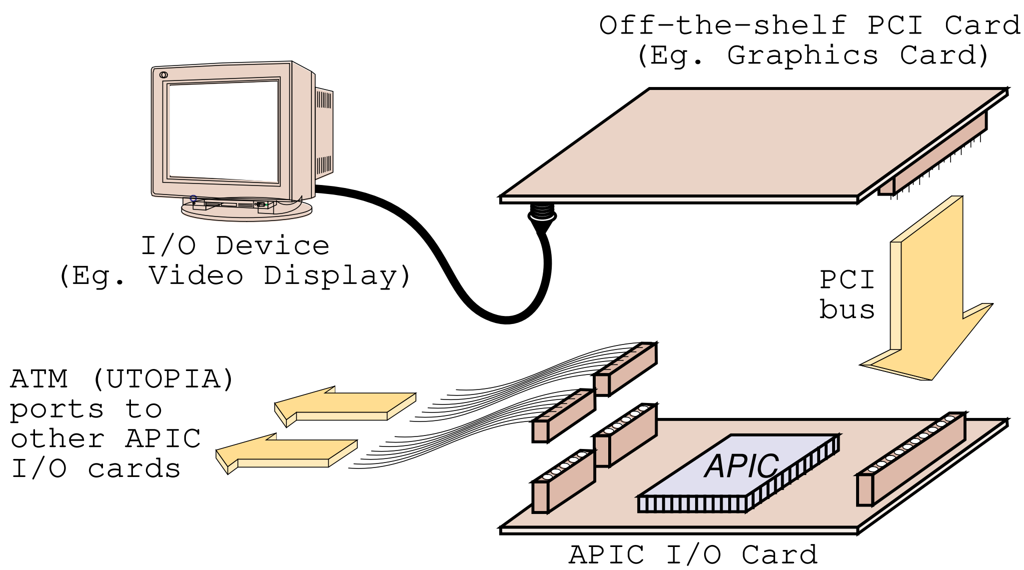

Figure 2: Interfacing Devices to the APIC

residing as a separate "device" on the interconnect. This method would work especially well when the processing device is an encryption/decryption engine, for example. Data destined for the network could be made to "pass through" the encryption device on its way to the link interface. Similarly, data arriving from the network could be made to pass through this device so the data can be decrypted before being forwarded to the destination I/O device. The second way to support media processing in our design is to include a local signal processor along with every device that generates or consumes data that may need processing. This method suffers from the drawback that the processor cannot be easily shared by multiple devices. However, in some cases this would be the approach of choice, such as, for example, when the media processor is a hardware compression/decompression engine. This ensures that uncompressed data would not have to traverse the interconnect, which is important because the data rate of the uncompressed stream might be a significant fraction of, or more than, the maximum bandwidth of the interconnect.

Interfacing Devices to the APIC

We plan to interface devices to the APIC in two phases. In the first phase, shown in Figure 2, we will build APIC I/O cards that provide two sockets for the two UTOPIA ATM ports, and a PCI socket for an off-the-shelf PCI card that can interface to the I/O device of our choice. Since the APIC includes an on-chip PCI bus interface, an APIC I/O card will have nearly no extra glue logic. Ribbon cables will be used to build the daisy chained interconnect of APIC I/O cards. Unless the off-the-shelf PCI device card has an on-board processor, there will be no local processing power associated with each device.

In the second phase, we plan to build cards that integrate the functions of both the APIC I/O card and the off-the-shelf PCI device card into a single card. This integration will allow us to directly interface both the APIC and the I/O device to the serial access port of an on-board VRAM, and thereby achieve much higher throughputs than would be possible with the configuration in Figure 2. Also, encryption and compression/decompression cards will be built in this phase.

Control of I/O Devices Using the APIC

As mentioned earlier, it is important that remote APICs and the devices connected to these APICs be controllable from anywhere in the network. This would avoid the need for dedicating a control processor (and other associated logic such as memory, PROMs, etc.) for every device residing in the network. This is especially true for small devices that perform very simple tasks, but still need to be on the network.

For instance, we may want to be able to control the brightness of a light bulb remotely, but it may not make economic sense to dedicate a control processor to every light bulb on the network.

The APIC itself can be controlled in two ways: either using memory-mapped I/O over the APIC bus interface, or through the use of control cells. Any APIC control operation that can be performed using memory-mapped I/O can also be done using a control cell. Thus, a processor can use memory-mapped I/O to control a directly connected APIC, and use that APIC to send control cells to remote APICs. To deal with the possibility of loss in the network, remote APICs always generate a response cell for every control cell that they process. Response cells are addressed to the source of the initiating control cell; this addressing is achieved by including the connection ID (VPI/VCI) to be used in the response cell header as part of every control cell. Furthermore, an alternating bit protocol is used to impose at-most once semantics on every control operation. Thus, regardless of whether a control cell or its response is lost, we are guaranteed that retransmissions of a control cell that did not get acknowledged by a response cell will not cause the control operation to be repeated.

To control remote devices, the APIC design allows for a different type of control cell. This type of cell will not cause a control operation in the remote APIC itself, but instead results in the remote APIC reading/writing from/to its locally connected external memory interface. In case of a WRITE operation, the data to be written, and the address at which it should be written are included in the control cell. The response cell merely serves to acknowledge successful completion of the WRITE operation. For a READ operation, the control cell contains the address from which data has to be read, and the response cell will carry the data back to the source. These READ/WRITE control cells can be used to read and write remote device registers if those device registers are mapped into the address space of the remote APIC. In a typical remote device control operation, the controlling processor would use memory-mapped I/O to command its directly connected APIC to send a control cell to the remote APIC; this control cell would result in the remote APIC using memory-mapped I/O to read or write the specified device registers on the remote device (which is directly connected to the remote APIC). Results of the control operation will be returned to the controlling processor in a response cell.

There is one more aspect to remote controllability that we haven't yet discussed, and this has to do with interrupts. What happens when a remote device wants to generate an interrupt to its controlling processor? One way to deal with this problem is to say that devices can never interrupt a remote processor, and require the remote processor to poll the device to detect the occurrence of an event. However, this places an unnecessary burden on the controlling processor, and all the advantages of interrupt driven programming are lost. The APIC solves this problem by introducing the concept of an "interrupt connection." This is a normally suspended transmit connection which has to be configured by the controlling processor in such a way that one or more "interrupt cells" are always queued on the connection, awaiting transmission. This connection remains suspended until some event occurs that requires the remote controlling processor to be interrupted. At this point, the interrupt connection is resumed, so that interrupt cells are transmitted. If the controlling processor had initialized the connection ID (VPI/VCI) field in the interrupt cells correctly, they will reach the APIC directly connected to the controlling processor, and that APIC would then generate a (normal) interrupt to the processor. By configuring the pacing rate of the interrupt connection, the control processor can make the remote APIC issue interrupt cells periodically, so that even if some are lost, one will eventually make it. When the control processor is interrupted, it uses a READ control cell operation to clear and acknowledge the interrupt from the remote device. When the interrupt gets acknowledged, the remote APIC automatically suspends the interrupt connection.

Due to space restrictions for this extended abstract, we have chosen not to elaborate on the software issues that arise in the control of the various devices. In brief, the operating system running on the controlling processor posts control cells to the mailbox of remote APICs, and returning responses are gathered by the system as they arrive, and processed in batches. The full version of the paper will include details of the kernel software libraries that we are building to support these control mailboxes.

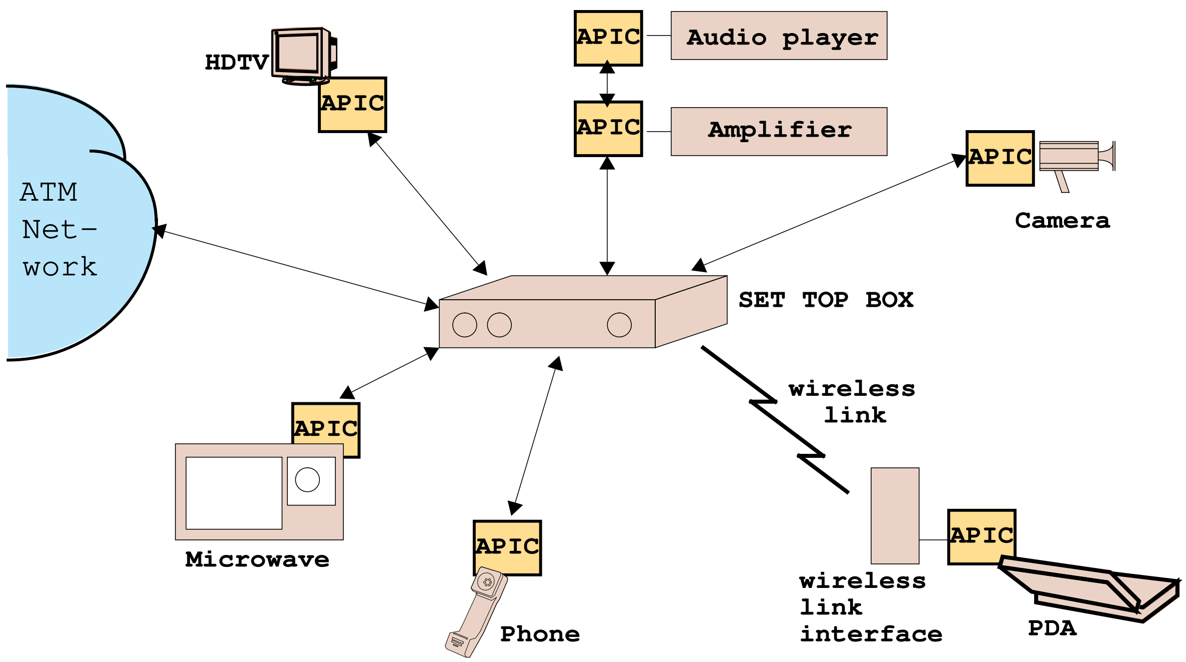

Figure 3: Architecture of a Home Area Network

Conclusions

Figure 3 shows how the APIC can be used to build a home area network. The APIC links to the set-top box (which contains the central control processor) can either be connected internally so that the overall topology is a star-shaped ring, or to an ATM switch which can be used to route the cells. Two aspects of the APIC are critical for its functioning in such an environment. The first is remote controllability, which was discussed in some detail in the preceding section. The second has to do with multirate access, and multirate pacing. Many devices do not require high bandwidth access to the network, and the APIC handles this by allowing for links of widely varying speeds, and the bus interface to the device can also run at speeds ranging from a few Kb/s to 2.4 Gb/s (bidirectional). In order to prevent fast senders from swamping slow receivers, the APIC can pace cells on a per-connection basis, at rates varying from a few Kb/s to 1.2 Gb/s. The expanded version of the paper will contain more details on the pacing and link interface design. In addition, we will expand on the following aspects of our design in the full length paper:

- The rationale behind making all devices “talk” ATM.
- The cost/flexibility/performance implications of using the APIC (or APIC-like) chip as a universal interface for a wide variety of devices.
- Details of the internal design of a set-top box that interfaces to APIC devices in the home.
- The software system architecture that is needed to make efficient use of the APIC interconnect architecture.

We expect the first APIC chips to return from the foundry in September of 1996. We have already begun the design of printed circuit boards that will allow the APIC to interface to workstations and a variety of I/O devices. We will also be building a large scale media server [2] by connecting individual disks to APICs on a chain, and striping video data over these APICs. There are also plans to build a high performance gigabit IP router [4] using APICs and a gigabit ATM switch [9] that has also been locally developed at Washington University.

References

[1] "An ATM PHY Data Path Interface: Draft Version 0.5", an ATM Forum Document Draft, May 1993.

[2] Buddhikot, Milind M.; Parulkar, Guru; and Cox, Jerome, R.; "Design of a Large Scale Multimedia Server," *Journal of Computer Networks and ISDN Systems*, Elsevier (North Holland), pp. 504-524, Dec 1994.

[3] Dittia, Zubin D.; Cox, Jerome R.; and Parulkar, Guru M.; "Design of the APIC: A High Performance ATM Host-Network Interface Chip," *Proc. IEEE INFOCOM '95,* Boston, 1995, pp. 179-187.

[4] Parulkar, Guru M.; Schmidt, Douglas C.; and Turner, Jonathan S.; "IP/ATM: a Strategy for Integrating IP with ATM," *Proc. ACM SIGCOMM '95*, August 1995, pp. 49-58.

[5] "Peripheral Component Interconnect (PCI), Revision 1.0 Specification", Intel Corporation, June 22, 1992.

[6] Hayter, Mark D.; and McAuley, Derek R.; "The Desk Area Network," *Operating Systems Review*, Vol. 25 No. 4, October 1991.

[7] Houh, Henry H..; Adam, Joel F.; Ismert, Michael; Lindblad, Christopher J.; and Tennenhouse, David L.; "The VuNet Desk Area Network: Architecture, Implementation, and Experience," to appear in *IEEE JSAC*, First Quarter, 1995.

[8] Leslie, Ian M.; McAuley, Derek R.; and Tennenhouse, David L.; "ATM Everywhere?," *IEEE Network*, Vol. 7 No. 2, March 1993, pp. 40-46.

[9] Turner, Jonathan S.; "An Optimal Non-Blocking Multicast Virtual Circuit Switch", *IEEE INFOCOM '94*, June 1994, pp. 298-305.