Report Number: WUCS-01-18

2001-01-01

The Smart Port Card: An Embedded Unix Processor Architecture for Network Management and Active Networking

John D. DeHart, William D. Richard, Edward W. Spitznagel, and David E. Taylor

This paper describes the architecture of the Smart Port Card (SPC) designed for use with the Washington University Gigabit Switch. The SPC uses an embedded Intel Pentium processor running open-source NetBSD to support network management and active networking applications. The SPC physically connects between a switch port and a normal link adapter, allowing cell streams to be processed as they enter or leave the switch. In addition to the hardware architecture, this paper describes current and future applications for the SPC.

The Smart Port Card: An Embedded Unix Processor Architecture for Network Management and Active Networking

John D. DeHart, William D. Richard,
Edward W. Spitznagel and David E. Taylor

WUCS-01-18

August 2001

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

The Smart Port Card: An Embedded Unix Processor Architecture for Network Management and Active Networking

John D. DeHart, William D. Richard, Edward W. Spitznagel, David E. Taylor

WUCS-TM-01-18

August 2, 2001

Department of Computer Science
Applied Research Lab
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130

Abstract

This paper describes the architecture of the Smart Port Card (SPC) designed for use with the Washington University Gigabit Switch. The SPC uses an embedded Intel Pentium processor running open-source NetBSD to support network management and active networking applications. The SPC physically connects between a switch port and a normal link adapter, allowing cell streams to be processed as they enter or leave the switch. In addition to the hardware architecture, this paper describes current and future applications for the SPC.

1 Introduction

The Washington University Gigabit ATM Switch [1] serves as a platform for research in several areas, including network management and active networking [2, 3]. As part of these research efforts, a new "Smart" Port Card (SPC) has been developed for use with the switch. The current Gigabit Switch has eight (8) ports capable of supporting link adapters operating at rates up to 2.4 Gb/s. Current link adapters available for use with the switch include a dual 155 Mb/s OC-3 SONET [4] link adapter, a 622 Mb/s OC-12 SONET link adapter, a 1.2 Gb/s Hewlett Packard (HP) G-Link [5] link adapter, and a dual 1.2 Gb/s HP G-Link adapter. The SPC physically connects between a switch port and a normal link adapter.

The purpose of a normal link adapter for an ATM switch is to provide the parallel-to-serial, encoding, and optical-to-electrical conversions necessary for data transmission over fiber using one of the optical transmission standards, e.g., SONET. Since no processing, other than encoding for transmission, can be done on a particular cell stream, link adapters are considered to be "dumb". The Smart Port Card adds processing capability at each switch port via the use of an embedded Pentium processor [6] running a NetBSD [7, 8] UNIX kernel. The architecture of the SPC allows cells to either transit through the SPC without modification or be written into the on-board memory for processing by the embedded Pentium processor. After processing, ATM cells can be read from on-board memory and transmitted either to the link adapter or to the switch.

The availability of a processor at each switch port allows the Gigabit Switch to process ATM cell streams on a per connection basis. This capability is being exploited by several research efforts that are currently under way at Washington University in St. Louis.

This paper describes in detail the architecture of the SPC. It also describes the steps required to construct the required embedded NetBSD kernel and fake BIOS. Network management and active networking research projects that are currently using the Smart Port Card are also outlined.

2 System Hardware Architecture

The architecture of the SPC is shown in Fig. 1. ATM cells that normally pass from the switch to the link adapter pass through an ATM Port Interface Controller (APIC) chip [9, 10] when the SPC is installed on a switch port. The APIC, when configured as a bus master, can write selected cell streams coming from either the switch or the link adapter directly into local memory via its PCI Bus [11] port. The APIC can also read cells from local memory and transmit them either to the switch or to the link adapter via the PCI bus as a bus master. Thus, the embedded Pentium processor is not responsible for cell transfer.

The SPC uses an embedded 166 MHz Intel Pentium MMX processor [6] running a NetBSD UNIX kernel [7] to implement network management functionality and to support active networking research. The processor, cache module, and north bridge chip [12] reside on a 3"x4" Intel embedded module [13] that connects to the SPC planar via two connectors. The PCI bus from the north bridge passes to the SPC planar via one of the connectors, and the other connector is used to pass the memory bus to the SPC planar. The current SPC supports up to 64 MBytes of EDO DRAM via a Small-Outline (SO) DIMM socket on the bottom of the SPC planar. The SPC measures 4.15"x7.95".

In a normal Pentium computer system, a south bridge chip [14] connects to the PCI bus and implements the system timer, interrupt controller, ISA bus interface, etc., as shown in Fig. 2. In the SPC, the required south bridge functionality is implemented using a Xilinx XC4020XLA-08 Field Programmable Gate Array (FPGA) [15]. This "System FPGA" also contains a small boot

ROM and a dual UART interface. The features of a normal computer system that are not needed in the embedded system are not implemented in the System FPGA, e.g., the ISA bus interface, the keyboard port, the mouse port, etc. The System FPGA does, however, provide an interface to a dual UART chip that is mapped to the memory addresses associated with COM0 and COM1 in the NetBSD kernel. The availability of these two COM ports has been extremely useful for diagnostic and debug purposes.

Via the development of the System FPGA, the functionality of an entire Pentium motherboard was reduced for embedded purposes to a small Intel processor module, an SO-DIMM socket, and a single FPGA. While other embedded processor architectures could have been used, the Intel architecture was chosen in order to support open-source NetBSD due to its availability and pervasive use in ATM network research at Washington University in St. Louis.

3 System FPGA

The System FPGA emulates the south bridge functionality of a Pentium motherboard. Due to the limited area on the SPC, an FPGA was developed to provide the minimal set of functionality required by the NetBSD kernel when running on an Intel Pentium architecture. A standard Pentium motherboard includes an 82371SB PIIX3 PCI ISA IDE Xcelerator (south bridge) chip [14], a Super I/O chip [16], a Flash memory BIOS chip, and both PCI and ISA connectors. Throughout the System FPGA design cycle, each of these devices was examined for functions and modes extraneous to the SPC. Only functionality essential to current and predicted SPC operation is implemented in the System FPGA.

As shown in the block diagram in Fig. 3, the System FPGA is a simple Slave PCI device with a PCI bus interface, a Register Manager, a Programmable Interval Timer (PIT), a Programmable Interrupt Controller (PIC), a Real-Time Clock Control (RTC) module, a Reset Control module, a 16-word BIOS module, and a UART Interface. The PCI Slave bus interface contains the state machines that decode and drive the PCI bus signals for I/O reads, writes, and interrupts. The Register Manager acts as an interface between the internal modules and the PCI Slave bus interface. It decodes addresses and routes data to and from the appropriate device.

In keeping with the PIIX3 (south bridge) architecture, the System FPGA provides a PIC that emulates two cascaded 82C59 [17] interrupt controllers. Since the embedded system only requires four interrupt request lines, i.e., one for the APIC, one for the PIT, and one for each COM port, only the master controller is fully implemented. In order to simplify the design while providing the necessary functionality, the PIC implements a static, fully-nested interrupt priority structure. Interrupt service routines must also use Specific End of Interrupt (EOI) commands exclusively as no other EOI commands are supported. Other modes and functions such as Special Mask Mode, Automatic End of Interrupt, and Priority Rotation are not currently supported.

The System FPGA contains a PIT for the generation of a periodic timer interrupt to the processor. While the PIIX3 contains three timers which are together functionally identical to an 82C54 programmable interval timer, only the system timer (counter 0) is emulated by the System FPGA since no refresh request signal (counter 1) or speaker tone (counter 2) is needed. Writes to the timer control word register in the PIT specify the rate at which timer interrupts are generated. Rate generation mode is currently the only supported mode. In order to emulate the 1.193MHz counter clock, the PIT has a finite-state machine that divides the 33MHz PCI clock to the System FPGA in order to achieve the desired average clock rate.

The RTC in the System FPGA, normally found in the super I/O chip on a Pentium mother board, contains a basic set of registers for time and date. Alarm registers exist for initialization

purposes only. For design simplification, they are implemented as simple read/write registers that do not generate alarms.

The System FPGA also contains a Reset Control module which drives internal reset during power-up and initialization. The SPC may also be reset via an ATM control cell sent to the APIC that initiates a write to a specific memory address decoded by the System FPGA. This allows the SPC to be completely reset remotely.

The System FPGA also provides an interface between the dual UART chip and embedded Pentium module. This provides two serial COM ports on the SPC for system monitoring and debug purposes.

The System FPGA also contains an internal 16-word by 32-bit "ROM" (this ROM is implemented using RAM bits inside the FPGA) that contains the code that is executed by the Pentium processor at power-up or reset. The program contained in the ROM monitors a specified memory location and waits for its contents to change from its initial default value. When this location changes, the program vectors execution to another predetermined memory location where a fake BIOS has been loaded via ATM control cells from a remote host.

Each module in the System FPGA is coded in behavioral VHDL (VHSIC Hardware Description Language) [18, 19, 20, 21] and connected to the other the modules via a top-level structural VHDL netlist. The Mentor Graphics QuickHDL simulator [22] was used to simulate each module. The entire System FPGA was simulated by driving the PCI signals at the "pins" of the chip. The stimulus files emulated a PCI bus master driving the bus with I/O reads, I/O writes, and interrupt service bus cycles. Following extensive simulation, the design was synthesized using the Exemplar Logic Spectrum [23] synthesis tool. The Xilinx Alliance Series [24] backend tools were used to place and route the design in a Xilinx XC4020XLA-08 FPGA. This FPGA has 20,000 equivalent logic gates, and 72% of the available logic blocks were used.

The final System FPGA design meets PCI electrical and timing specifications and fully supports those Pentium motherboard features required by the Pentium processor when running NetBSD on the SPC.

4 Embedded NetBSD Implementation

The embedded NetBSD implementation consists of a custom NetBSD kernel with a memory disk and a "fake" BIOS which also acts as a fake boot loader. NetBSD version 1.4.1 is currently being used.

The custom NetBSD kernel for the SPC differs from a generic NetBSD kernel in that it uses a serial console instead of a VGA display and keyboard, a memory disk instead of a physical hard disk, and the APIC device driver instead of a more standard network interface device driver. The APIC device driver used for SPC implementation is the same driver used with an APIC Network Interface Card (NIC) in a normal PC. These features are specified in the kernel configuration file; modification of the kernel source code itself is very minimal for use with the SPC.

In the boot process, the fake BIOS and the NetBSD kernel are loaded into system memory via ATM control cells, sent from an external host and processed by the APIC on the SPC. The ROM inside the System FPGA holds a small program that waits for a predetermined memory location to change value. This program is run at power-up. Once the fake BIOS and the NetBSD kernel are loaded, another ATM control cell is used to change the predetermined memory location, signaling the processor to jump to the first instruction of the fake BIOS.

The fake BIOS is an assembly language program that performs some of the actions which are normally done by the BIOS on the Pentium motherboard upon power-up and some of the actions

that are normally done by the NetBSD boot loader before kernel execution takes place. Specifically, the fake BIOS initializes registers inside the north bridge indicating the type and size of memory in the system, puts the processor into protected mode, and sets up the data structures the NetBSD kernel expects. After initializing the system, the fake BIOS jumps to the entry point in the NetBSD kernel.

Since the SPC has no physical disk, the NetBSD kernel uses a memory disk. The filesystem for the memory disk is added to the kernel image after the kernel itself is built. At a minimum, the file system must contain the /sbin/init program for the system to boot. Additional programs can be added as needed for SPC applications, as described below, and for testing purposes.

5 Network Management and Active Networking Applications

The SPC provides a flexible platform for numerous networking applications and has become an integral part of several research projects at Washington University in St. Louis. The first project is the construction of a scalable, high performance active network node. Active networks are packet-switched networks in which the packets can contain code fragments, or references to code fragments, that are to be executed on the intermediary nodes in the network. One goal of active network research is to develop mechanisms for network customization and increased flexibility. End systems can then inject code into the network to change the network's behavior to suit their needs.

The SPC is being used as one component in a scalable active network node. The processing power of the SPC gives the node the ability to do active processing at each port of a network switch. Software running on the SPC can perform functions such as packet classification, packet scheduling, routing, active module loading and active module invocation. The combination of the Washington University Gigabit ATM Switch populated with SPCs at each port and the software environment being developed in the active network research project will yield a high performance active network node with a target of active networking at gigabit rates.

Applications of active networking include congestion control, network protocol prototyping and deployment, encryption, and application specific computation. Many of these will be demonstrated in a testbed network that is initially comprised of three gigabit ATM switches populated with SPCs. Fig. 4 shows a three switch testbed configuration that has been used to demonstrate active congestion control for WaveVideo [25], the first of our active networking applications. In this demonstration, three video displays are used to show the effects of traffic congestion on 1.3 and 2.6 Mb/s video streams with and without active processing. The WaveVideo application encodes the video into 33 different frequency sub-bands. With no congestion control, packets are discarded randomly and may be from any of the frequency sub-bands. Under high cross-traffic conditions, the resulting video is unusable without active processing. The active processing performed in the SPCs causes the higher frequencies to be discarded first when congestion is detected. The active code is downloaded from the Code Server on-demand and dynamically loaded into the SPC kernels when the video stream is initiated. The resulting video is of significantly better quality and is usable under much higher congestion levels than is the video with random packet discarding. Many of the details of the experimental configuration used in this demonstration can be found in [26].

A second research project using the SPC is the construction of a highly scalable network monitoring, visualization and control system. When a network event such as an outage or link congestion causing loss of data occurs, it is not always clear to the end user or to the network operator where the problem exists. Network events are also often transient and thus hard to isolate. Polling in order to ascertain the health of the network introduces extra traffic which may in fact add to the problem. What is needed is an automatic system for detecting network problems and for performing

control functions to alleviate them.

A key capability required in the network monitoring, visualization and control system is the collection of traffic data on a per-flow basis. It is crucial that this data collection take place in a non-intrusive manner so that any existing network problem is not intensified. Per-flow data is important in order to be able to identify whether there is a particular flow that is violating its usage constraints. If such a flow is found, corrective measures should be applied to that flow first before other well-behaved flows are affected. Per-flow data might also be used in visualizations that would depict the performance and utilization experienced by different users of the network.

The SPC provides an excellent mechanism for providing such a non-intrusive probe. Software similar to the active networking software discussed above is being developed to provide the mechanisms for identifying and manipulating flows and per flow data. Fig. 5 shows a network configuration that would use the SPC as a network probe in a network monitoring, visualization and control system.

6 Discussion

The overall SPC project included many new developments:

- APIC chip
- APIC Network Interface Card (APIC NIC)
- System FPGA
- SPC Development System (SPCDS)
- SPC circuit board
- Embedded NetBSD kernel

Each part of the SPC project posed its own set of issues and design challenges. The APIC NIC, a PCI card based on the APIC chip, was first developed for use in a normal PC along with a NetBSD driver. The Smart Port Card Development System (SPCDS) was developed next as a flexible platform for SPC development.

As a prototype platform, the SPCDS coupled with a commercial PCI bus analyzer [27] proved invaluable in understanding and developing the System FPGA and the embedded NetBSD kernel. The SPCDS also provided a means for testing board level issues such as how the APIC operates in the SPC environment. As shown in Fig. 6, the SPCDS contains two PCI bus slots which were used to connect an APIC NIC and the PCI bus analyzer to the SPCDS PCI bus during testing. The combination of an SPCDS and an APIC NIC provided all of the functionality of the final SPC along with the ability to use the PCI bus analyzer for observing PCI bus cycles. Since the final SPC has no PCI slots, use of the PCI bus analyzer with the SPC is not possible.

APIC NICs were first available starting in April of 1999. The system FPGA design was completed in May of 1999. The first successful boot of the embedded NetBSD kernel on the SPCDS occurred in June of 1999. After completing initial testing using the SPCDS, 25 SPCs were constructed in August of 1999. Photographs of the final SPC are shown in Fig. 7.

Fifty-five additional SPCs are currently being constructed for distribution to participants of Washington University's Gigabit Network Technology Distribution Program [28]. As part of this program, which is supported by the National Science Foundation, 50 gigabit switches have been

constructed and distributed to 30 universities around the world. Each of these kits includes six (6) APIC NICs.

A new version of the SPC is being considered that would use a faster processor. Addition modifications that could be implemented include the use of FPGAs in the data paths between the APIC and the link adapter and the APIC and the switch core for hardware-based cell processing. Other researchers are also working on similar techniques [29].

7 Conclusions

By current standards, the 166 MHz Pentium processor used on the SPC is slow, and moving to a faster and more capable processor would significantly increase the capability of the Smart Port Card. Moving from a 33 Mhz, 32-bit PCI bus to a faster 66 MHz, 64-bit bus would also increase the capability of the SPC significantly by providing four (4) times the system memory bandwidth. Even with these limitations, the SPC in it's current form has been found to provide an excellent platform on which to study network management and active networking applications. The hardware approach described here does allow the necessary functionality of a complete Pentium mother board required by NetBSD to be integrated onto the SPC. NetBSD also provides an excellent OS for this type of application.

Acknowledgments

This work was supported the National Science Foundation (grant ANI-9616754) and by the Defence Advanced Research Projects Agency (contract N66001-98-C-8510).

References

[1] Chaney, Thomas J., Fingerhut, Andrew, Flucke, Margaret, and Turner, Jonathan S. *Design of a Gigabit ATM Switch*, Proceedings of INFOCOM'97, April 1997.

[2] Decasper, D., Parulkar, G., Choi, S., DeHart, J., Wolf, T., Plattner, B. *A Scalable, High Performance Active Network Node*, IEEE Network, January/February 1999.

[3] Parulkar, G., Schmidt, D., Kraemer, E., Turner, J., Kantawala, A. *An Architecture for Monitoring, Visualization and Control of Gigabit Networks*, IEEE Network, September/October 1997.

[4] ANSI T1.106-1988, *Telecommunications - Digital Hierarchy Optical Interface Specifications: Single Mode*, 1988.

[5] HDMP-1022 Transmitter/HDMP-1024 Receiver Data Sheet, Hewlett-Packard Corporation, 1997.

[6] Pentium Processor Family Developer's Manual, Intel Corporation, Mt. Prospect, IL, 1997.

[7] NetBSD, http://www.netbsd.org

[8] McKusick, Marshall Kirk; Bostic, Keith; Karels, Michael J.; Quarterman, John S. *The Design and Implementation of the 4.4 BSD Operating System* Addison Wesley, 1996.

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

[10] Dittia, Z., Parulkar, G., Cox, J. *The APIC Approach to High Performance Network Interface Design: Protected DMA and Other Techniques*, Proceedings of INFOCOM'97, April 7-11, 1997, Kobe, Japan.

[11] PCI Local Bus Specification, Rev. 2.1, PCI Special Interest Group, Portland, OR, 1995.

[12] Intel 430HX PCISET 82439HX System Controller (TXC) Data Sheet, Intel Corporation, San Jose, CA, April, 1997.

[13] Embedded Processor Module Design Guide, Intel Corporation, Mt. Prospect, IL, 1997.

[14] 822371FP (PIIX) and 82371SB (PIIX3) PCI ISA IDE Xcelerator Data Sheet, Intel Corporation, San Jose, CA, 1997.

[15] Xilinx 1999 Data Book, Xilinx, Inc., San Jose, CA, 1999.

[16] PC87323VUL Super I/O Chip Data Sheet, National Semiconductor Corporation, 1995.

[17] Microprocessor and Peripheral Handbook, Vol. 1, Intel Corporation, Santa Clara, CA, 1987.

[18] Standard 1076.3, *IEEE Standard VHDL Language Reference Manual*, IEEE, 1994.

[19] Standard 1164-1993, *IEEE Standard Multivalue Logic System for VHDL Model Interoperability*, IEEE, 1993.

[20] Standard 1076.3, *VHDL Synthesis Package*, IEEE, 1995.

[21] Standard 1076.A, *VITAL ASIC Modeling Specification*, IEEE, 1995.

[22] Getting Started with QuickHDL and VHDLwrite, Mentor Graphics Corporation, Portland, OR, Nov. 1996.

[23] Exemplar Leonardo Spectrum Level 3 Ver. 1999.1d Manual, Exemplar Logic, Alameda, CA, 1999.

[24] Xilinx Alliance Series Software Ver. 1.5i Manual, Xilinx Corporation, San Jose, CA, 1998.

[25] G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, and B. Stiller, *WaveVideo An Integrated Approach to Adaptive Wireless Video*, ACM Monet, Special Issue on Adaptive Mobile Networking and Computing, Vol. 4, No. 4, 1999.

[26] Keller, R., S. Choi, M. Dasen, D. Decasper, G. Fankhauser, and B. Plattner, *An Active Router Architecture for Multicast Video Distribution*, Proceedings of Infocom 2000, March 2000, Tel Aviv, Israel.

[27] VMETRO PBT-415 PCI Bus Analyzer & Exerciser User's Manual VMETRO Inc., Houston, TX, 1998.

[28] http://www.arl.wustl.edu/gigabitkits/kits.html

[29] I. Hadzic and J. Smith, *On-the-fly Programmable Hardware for Networks*, Proc. GLOBECOM 1998.

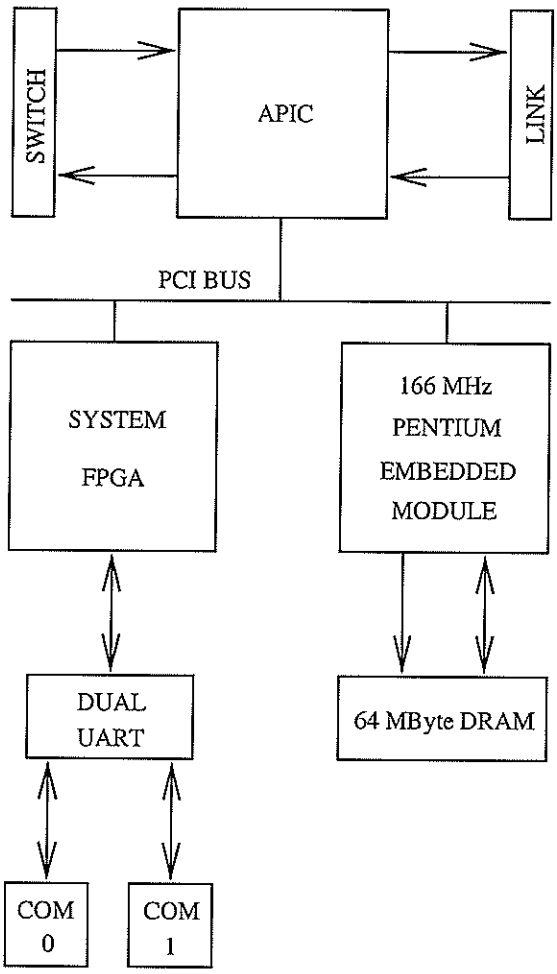

Figure 1: *Block diagram of Smart Port Card (SPC) architecture. The SPC uses an Intel 166MHz Pentium Embedded Module, a Xilinx XC4020XLA-08 Field Programmable Gate Array (FPGA), 64MBytes of EDO DRAM, a dual UART chip, and an ATM Port Interface Controller (APIC) to support an open-source NetBSD operating system. The System FPGA provides an essential set of South Bridge functionality in order to support NetBSD running on the Intel embedded module. The APIC is an ASIC which provides an interface between the switch port and line card capable of switching streams to and from local memory for processing.*

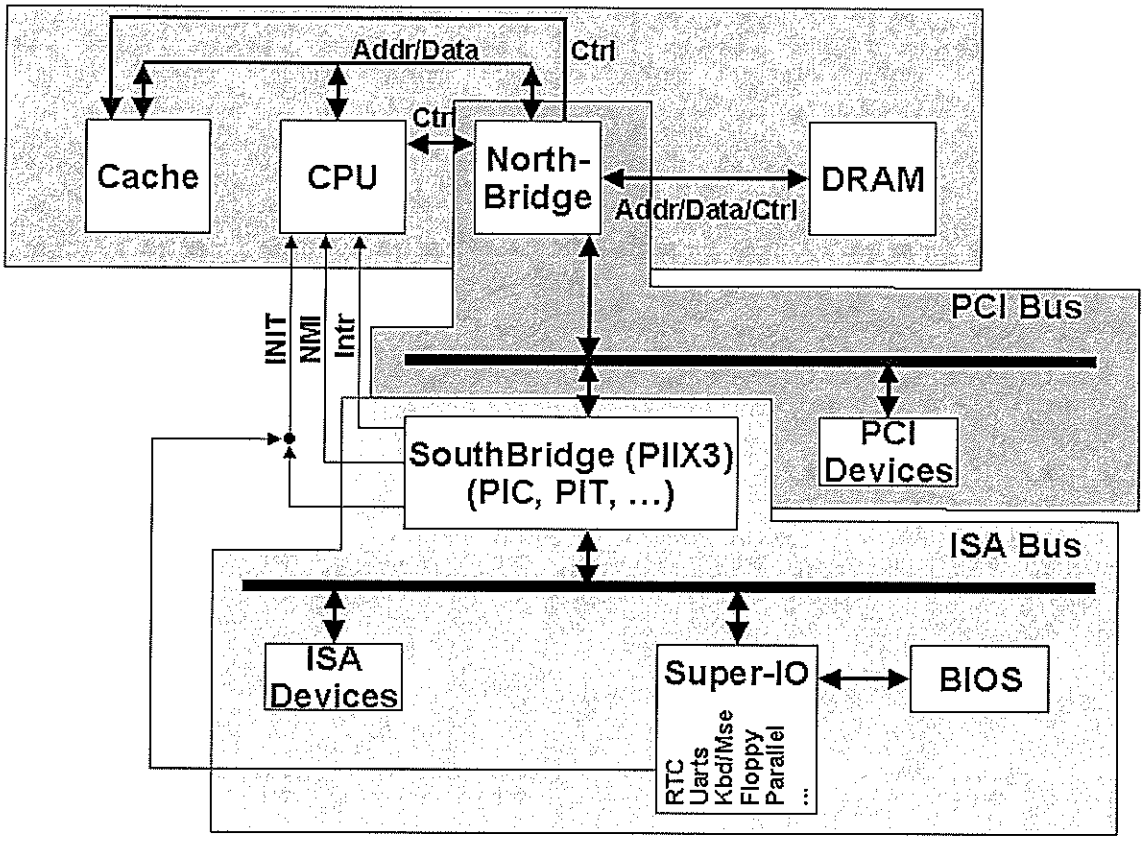

Figure 2: *Architecture of a Legacy Pentium PC. The PIIX3 south bridge "bridges" the PCI and ISA buses and supports, in conjunction with a super I/O chip, all of the legacy devices, e.g., the system timer, interrupt controller, floppy interface, parallel interface, UARTs, etc.*

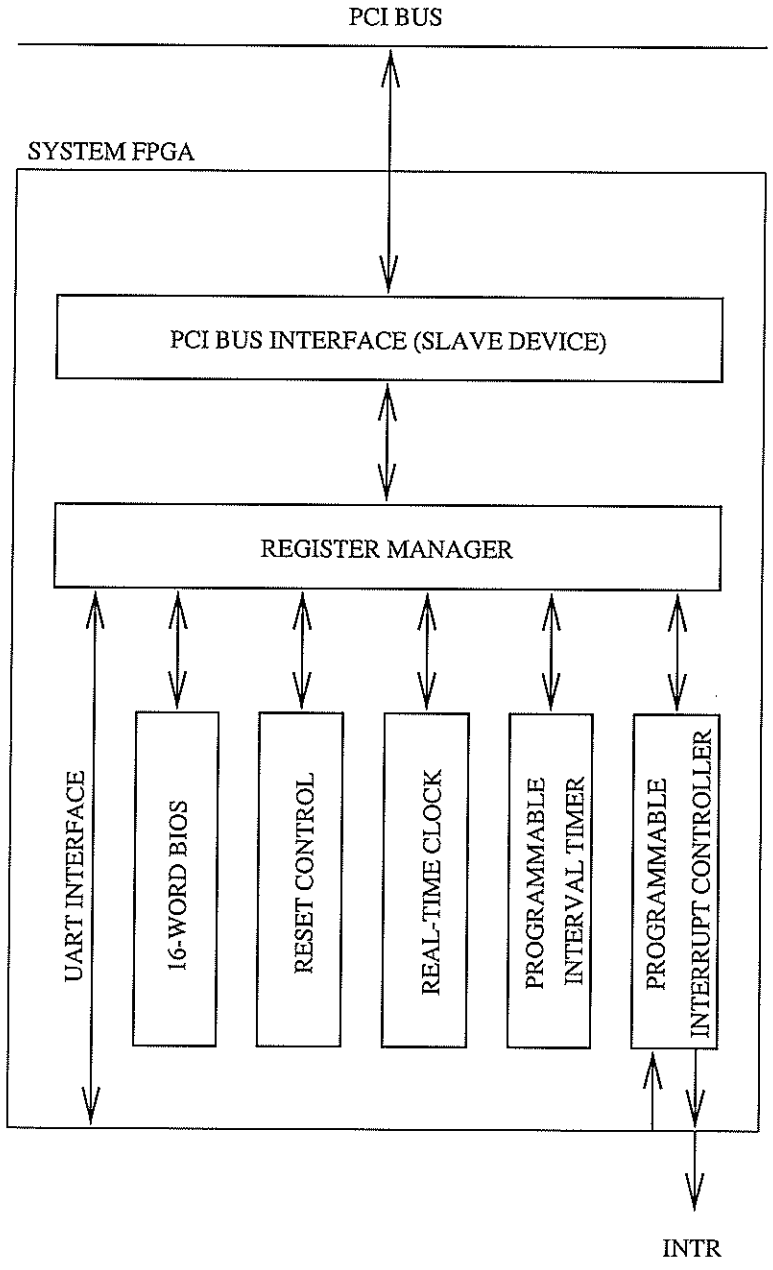

Figure 3: *Block diagram of System FPGA. Due to physical space constraints on the SPC, the System FPGA was designed to implement a minimal set of South Bridge functionality in order to support the Intel embedded module. Acting as a PCI slave device, the System FPGA provides a Programmable Interrupt Controller (PIC), Programmable Interval Timer (PIT), Real-time Clock, Reset Control, dual UART interface, and 16-word BIOS. The Register Manager provides an internal interface between each module and the PCI Slave bus interface.*

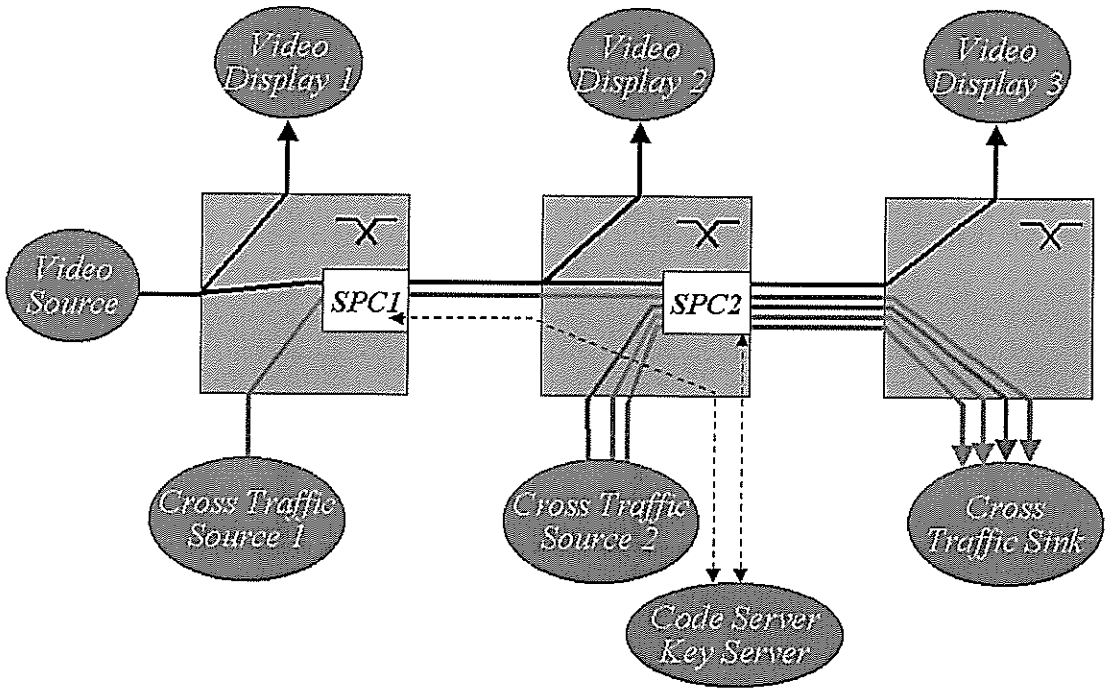

Figure 4: *Active Networking Multipoint Video Congestion Control. In this demonstration network, Active Routers populated with SPCs are used to show the application of active networking to congestion control. An active congestion control module is used by the Active Routers to process the video stream coming from the video source. By turning this active module on and off at each of the SPCs, the affect on the displayed video quality is clearly shown. The affect of congestion on the video traffic is also highlighted by varying the amount of cross traffic that impacts the three video displays.*

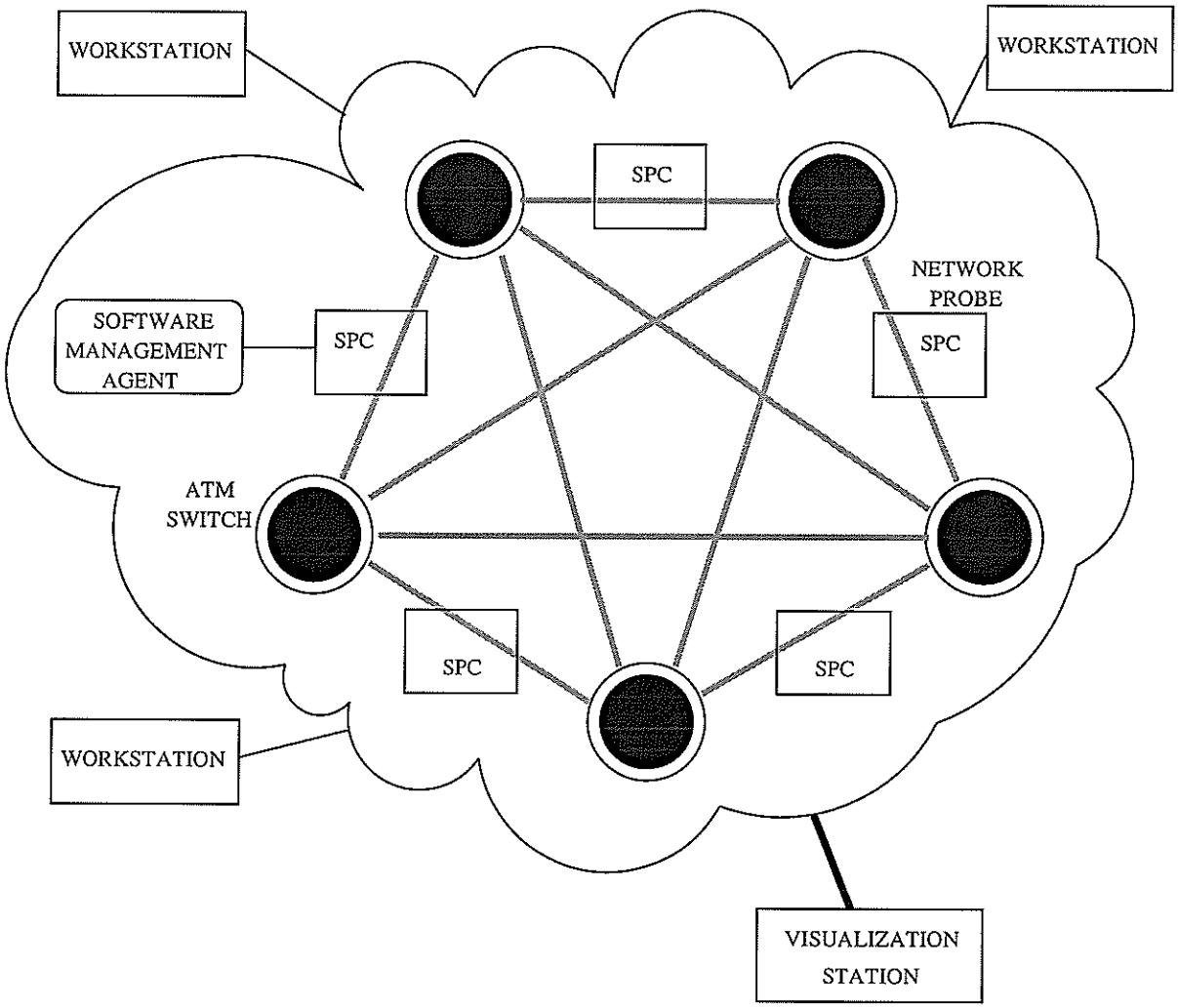

Figure 5: *Network Monitoring, Visualization and Control System. This network configuration illustrates a use of the SPC in a Network Monitoring, Visualization and Control System. The SPCs are used as network probes, gathering traffic data at different points in the network. The Visualization Station is used to display traffic visualizations that will make it easier for network managers to understand the operation of their network. The Software Management Agent is the entity that does local processing of data and event information before passing it on to a higher level management system or visualization station.*

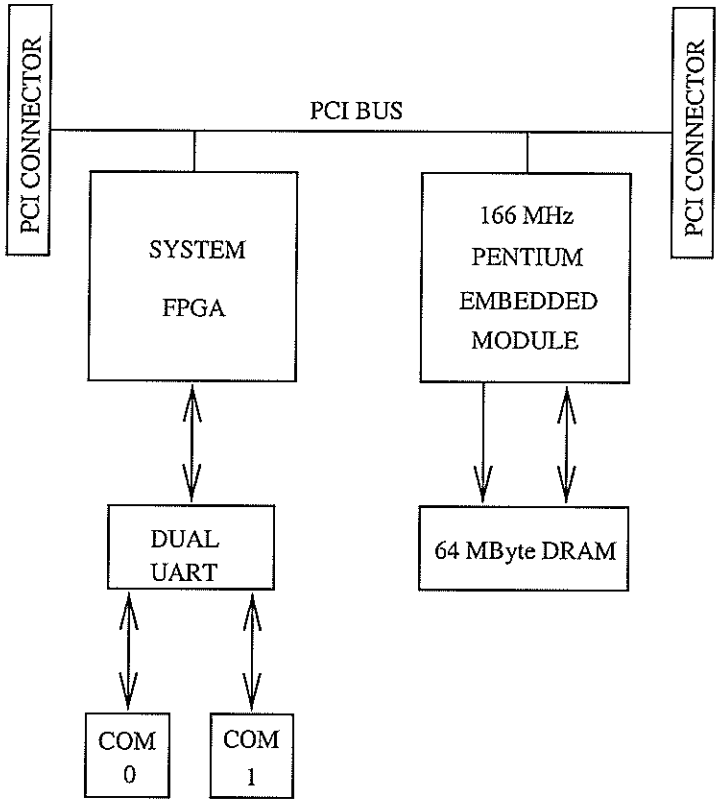

Figure 6: *Block diagram of Smart Port Card Development System (SPCDS). In addition to the embedded Pentium module, 64MB of DRAM, System FPGA, and Dual UART chip, the SPCDS contains two PCI bus slots which were used to connect an APIC NIC and the PCI bus analyzer to the SPCDS PCI bus during testing.*

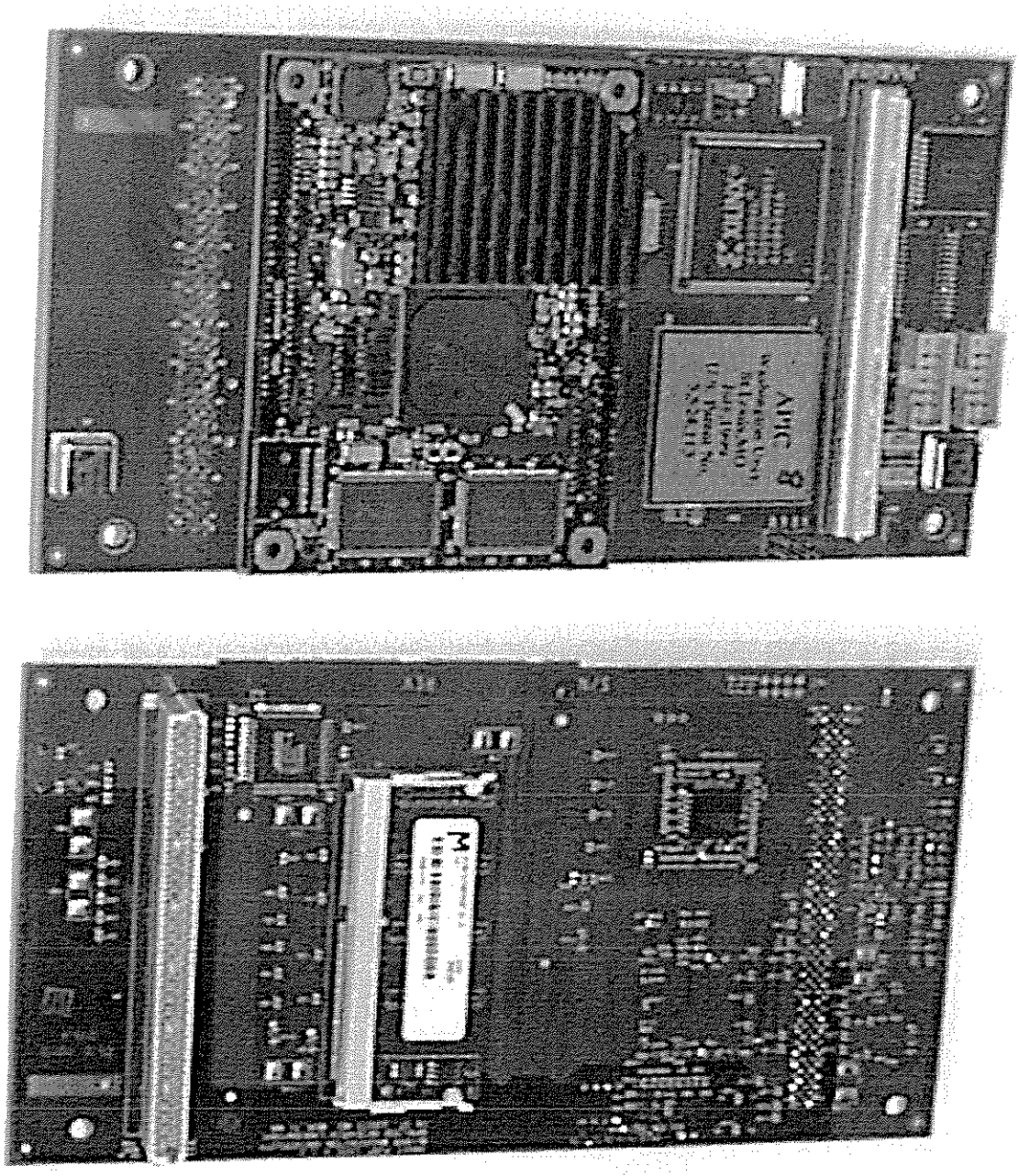

Figure 7: *Photographs of the Smart Port Card (SPC). The top photograph shows the top view of the SPC. The Intel embedded module, the Xilinx 4020XLA-08 FPGA, and the APIC chip are in the center of the card. The connector at the right connects to the link adapter. Two serial connectors are shown on the right edge of the card. The bottom photograph shows the bottom view of the SPC. The SO-DIMM memory module is shown in the left center of the card. The connector at the left connects to the Gigabit Switch.*