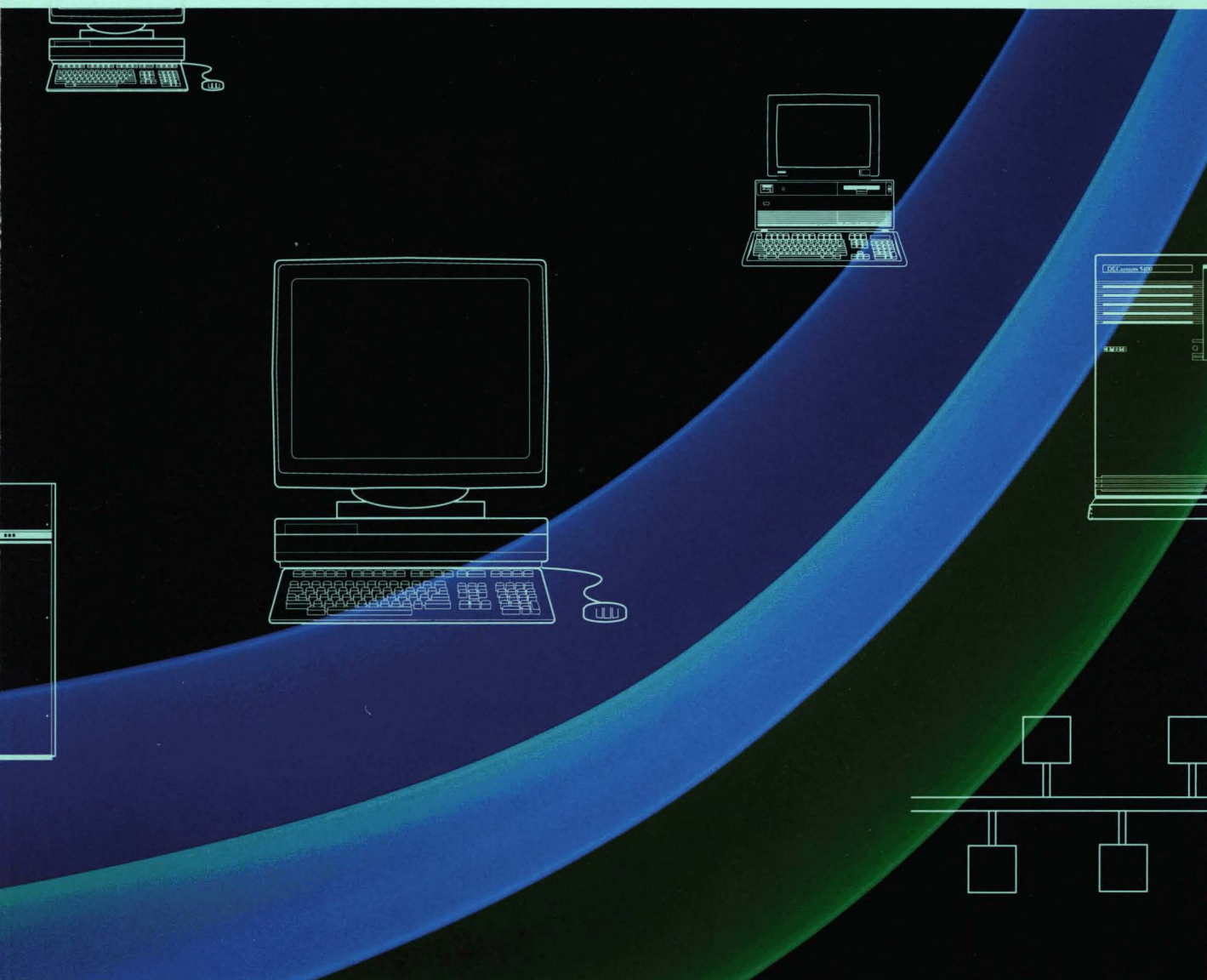


Digital Technical Journal

Digital Equipment Corporation



Cover Design

Our cover depicts the dual fiber-optic ring that is the physical medium for high-speed transmission of data in Digital's new FDDI local area network. High-performance workstations, systems, servers, and other LANs can be easily connected to the ring through new bridge and wiring concentrator products, which are among the topics featured in this issue on FDDI.

The cover was designed by Dave Bryant of Digital's Media Communications Group.

Editorial

Jane C. Blake, Editor
Kathleen M. Stetson, Associate Editor

Circulation

Catherine M. Phillips, Administrator
Suzanne J. Babineau, Secretary

Production

Helen L. Patterson, Production Editor
Nancy Jones, Typographer
Peter Woodbury, Illustrator

Advisory Board

Samuel H. Fuller, Chairman
Richard W. Beane
Robert M. Glorioso
Richard J. Hollingsworth
John W. McCredie
Alan G. Nemeth
Mahendra R. Patel
F. Grant Saviers
Robert K. Spitz
Victor A. Vyssotsky
Gayn B. Winters

The *Digital Technical Journal* is published quarterly by Digital Equipment Corporation, 146 Main Street MLO1-3/B68, Maynard, Massachusetts 01754-2571. Subscriptions to the Journal are \$40.00 for four issues and must be prepaid in U.S. funds. University and college professors and Ph.D. students in the electrical engineering and computer science fields receive complimentary subscriptions upon request. Orders, inquiries, and address changes should be sent to The *Digital Technical Journal* at the published-by address. Inquiries can also be sent electronically to DTJ@CRL.DEC.COM. Single copies and back issues are available for \$16.00 each from Digital Press of Digital Equipment Corporation, 12 Crosby Drive, Bedford, MA 01730-1493.

Digital employees may send subscription orders on the ENET to RDVAX::JOURNAL or by interoffice mail to mailstop MLO1-3/B68. Orders should include badge number, cost center, site location code and address. All employees must advise of changes of address.

Comments on the content of any paper are welcomed and may be sent to the editor at the published-by or network address.

Copyright © 1991 Digital Equipment Corporation. Copying without fee is permitted provided that such copies are made for use in educational institutions by faculty members and are not distributed for commercial advantage. Abstracting with credit of Digital Equipment Corporation's authorship is permitted. All rights reserved.

The information in this Journal is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this Journal.

ISSN 0898-901X

Documentation Number EY-H876E-DP

The following are trademarks of Digital Equipment Corporation: CI, DDCMP, DEC, DECbridge, DECconcentrator, DEC FDDI controller, DECnet, DECMCC, DECstation, DECWORLD, Digital, the Digital logo, DNA, LAT, ThinWire, TURBOchannel, ULTRIX, VAX, VAXcluster, VMS.

AppleTalk is a registered trademark of Apple Computer, Inc.

C is a registered trademark of Microsoft Corporation.

Motorola is a registered trademark of Motorola, Inc.

SUN is a registered trademark and NFS is a trademark of Sun Microsystems, Inc.

X Window System is a trademark of the Massachusetts Institute of Technology.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

Book production was done by Digital's Media Communications Group in Bedford, MA.

| Contents

- 8 **Foreword**
Mark F. Kempf

Fiber Distributed Data Interface

- 10 **Fiber Distributed Data Interface Overview**
William R. Hawe, Richard Graham, and Peter C. Hayden
- 19 **Development of the FDDI Physical Layer**
Jerry D. Hutchison, Christopher Baldwin,
and Bruce W. Thompson
- 31 **FDDI Data Link Development**
Henry S. Yang, Barry A. Spinney, and Stephen Towning
- 42 **An Overview of the Common Node Software**
Paul W. Ciarfella, David Benson, and David S. Sawyer
- 53 **Development of the DECbridge 500 Product**
Robert C. Kochem, James S. Hiscock, and Brian T. Mayo
- 64 **The DECconcentrator 500 Product**
William J. Tiffany, G. Paul Koning, and James E. Kuenzel
- 76 **DECelms—Managing Digital's FDDI and Ethernet
Extended Local Area Networks**
Bruce E. Sweet
- 85 **ULTRIX Fiber Distributed Data Interface
Networking Subsystem Implementation**
Ursula Sinkewicz, Chran-Ham Chang, Lawrence G. Palmer,
Craig Smelser, and Fred L. Templin

Editor's Introduction



Jane C. Blake
Editor

Digital's fiber distributed data interface, FDDI, is a high-speed LAN that links workstations, systems, and other local area networks such as Ethernet. This new LAN offers distance as well as speed, and extends connections beyond the limits of a building to cover wider areas such as a campus. Papers in this issue of the *Digital Technical Journal* provide insights into the technology choices made during FDDI development and describe the design of the layers and products that make up this 100-megabit/second LAN system.

Among the reasons Digital's engineers chose FDDI technology were its high speed, high throughput, and consistency with existing and evolving standards. In this issue's opening paper, Bill Hawe, Rich Graham, and Peter Hayden review these selection criteria and discuss the reasoning behind the final choices. The authors additionally present an overview of FDDI layers and product operations, which establishes a context for the papers that follow.

The lowest FDDI layer provides the physical connections for data transmission on the fiber-optic ring. In their paper, Jerry Hutchison, Chris Baldwin, and Bruce Thompson describe the operation of the physical layer, the functional partitioning, and the choice of chip set technologies. They then examine the distributed clocking scheme and present the methods used in the design of the optical link (methods later adopted by the Physical Layer Medium Dependent Working Group of the FDDI committee).

Both the development of the physical layer and the next higher layer, the data link layer, resulted in contributions to the ANSI FDDI standard. Digital's implementation of the data link layer is the subject of the next paper by Henry Yang, Barry Spinney, and Steve Towning. In addition to presenting several key algorithms, the authors review the functions of the three data link chips. They conclude their paper with a discussion of chip simulation and test.

Consistent behavior in the physical and data link layers is managed by a set of reusable software libraries called Common Node Software. Paul Ciarfella, Dave Benson, and Dave Sawyer relate the events that led to the development of CNS and describe its functions. CNS implements the protocols defined by the FDDI station management standard and manages the FDDI chip set. CNS is included in the DECbridge and DECconcentrator products.

The DECbridge 500 product is the bridge for traffic between the high-speed FDDI LAN and the slower Ethernet 802.3 LANs. Design considerations and DECbridge functions are presented by Bob Kochem, Jim Hiscock, and Brian Mayo. Their discussion offers insight into the complexities involved in connecting LANs with different data rates, frame formats, and frame sizes.

Bill Tiffany, Paul Koning, and Jim Kuenzel then describe the DECconcentrator 500 product. The DECconcentrator, the cornerstone of the FDDI LAN, provides additional ports to which stations can be connected by radially wired cables. The authors examine the significance of Digital's choice of a dual ring of trees topology, which led to the need for a concentrator, and give details of DECconcentrator development.

Remote management of the DECconcentrator and DECbridge products, as well as of Ethernet bridges, is provided by DECelms software. Bruce Sweet outlines the challenges DECelms developers faced, including an evolving ANSI FDDI standard and differences between FDDI and Ethernet technologies. He then describes the network management architecture and gives details of features that benefit the network manager.

The final paper in this issue addresses the development of an adapter that allows high-performance RISC workstations to connect to FDDI. Ursula Sinkewicz, Chran-Ham Chang, Larry Palmer, Craig Smelser, and Fred Templin review ULTRIX support for the FDDI system and then give details of the DEC FDDIcontroller 700 adapter, which provides a single FDDI attachment for DECstation 5000 workstations. The authors have included discussions of relevant performance data.

The editors thank Mark Kempf for his help in initiating and preparing this issue, and for kindly agreeing to write the issue's Foreword.

Jane Blake

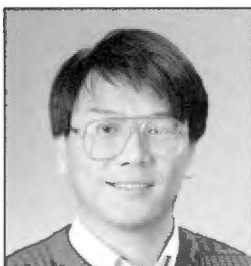
Biographies



Christopher Baldwin As a principal engineer in the Network Systems Engineering Group, Christopher Baldwin is responsible for the development of the fiber-optic physical layer standard for FDDI (PMD) and helped develop the fiber-optic hardware used in FDDI products. He also worked on fiber-optic transmission for the LAN Bridge 200. Before joining Digital in 1986, Chris was a senior engineer at Polaroid Corporation. He holds a B.A. (1983, honors) in electrical engineering from Brown University and an M.S. (1984) from the Institute of Optics at the University of Rochester.



David Benson Software principal engineer Dave Benson is a member of the Communications Systems Engineering Group and the project leader for the Common Node Software. His next responsibility will be as a firmware project leader in the area of future FDDI products. Previously, Dave designed and implemented the physical layer portion of CNS, codeveloped the design verification test (DVT) monitor tool, and was a member of the hardware design team for the FDDI tester. Dave came to Digital in 1986 with ten years' engineering experience from Honeywell Information Systems.



Chran-Ham Chang Chran-Ham Chang is a member of the ULTRIX Network Engineering Group. As a senior software engineer, he is responsible for the ULTRIX FDDI and Ethernet driver design and development. Earlier, Chran was involved in the analysis of ULTRIX network performance and in the design of performance tools. He joined Digital in 1987 after receiving his M.S.C.S. from New Jersey Institute of Technology. Prior to this, Chran worked as a software specialist in Taiwan for the distributor of Digital's products.



Paul W. Ciarfella As a software senior engineer in the Communications Systems Engineering Group, Paul Ciarfella is currently developing DECmcc software to manage FDDI networks. Paul was one of the developers of the Common Node Software. Previously, he performed simulation and design verification testing of Digital's FDDI chip set and developed system software for the FDDI tester. Paul joined Digital in 1987 after receiving a B.S.C.S. (high honors) from Northeastern University. He is a co-applicant for a patent related to increasing the robustness of FDDI networks.



Richard Graham Richard Graham is a consulting engineer in the Local Area Networks Business Unit. Since joining Digital in 1981, he has worked on a number of projects, including early Ethernet development, the Ethernet repeater, the Ethernet-to-Ethernet bridge (LAN Bridge 100), DEC Standard 134, and most recently, the FDDI-to-Ethernet bridge. Prior to this, Rich was a senior engineer in the Computer System Division of Harris Corporation. He holds a B.S. (1975) in electrical engineering and an M.S. (1977) in computer engineering, both from Syracuse University.



William R. Hawe Senior consulting engineer Bill Hawe manages the LAN Architecture Group. He is involved in designing new LANs and extended LAN technologies. He and his group also design portions of DECnet and TCP/IP architectures. While in Corporate Research, Bill worked on the design of the Ethernet with Xerox and Intel. Since joining Digital in 1980, Bill has done extensive performance analysis and has established this as an integral part of Digital's networking architecture. He holds a B.S.E.E. and an M.S.E.E. He has published 27 papers outside Digital and has 11 patents issued or pending.



Peter C. Hayden Peter Hayden is a principal engineer in the Telecommunications and Networks, Software Development Group. He joined Digital in 1986 as a member of the FDDI team and has contributed to the development of the FDDI technology and product set. He was instrumental in the development of the FDDI test platform which allowed for execution of test programs in both simulation and hardware environments. Peter also led the team that developed the FDDI control software resident in all FDDI products. He holds B.S.E.E. and M.S.C.S. degrees from Union College in Schenectady, NY



James S. Hiscock Principal software engineer James Hiscock is currently a firmware project leader for a future bridge/router product. Prior to this, he led the DECbridge 500 and LAN Bridge 200 firmware projects. He also worked on the UNIBUS-to-Ethernet adapter and the LAN Bridge 100 device. Before coming to Digital in 1984, Jim was employed as a software engineer in the air traffic control division of Raytheon Company. He received a B.S. (1982) in computer systems engineering from the University of Massachusetts.



Jerry D. Hutchison Jerry Hutchison joined Digital in 1977 after receiving B.S. and M.S. degrees in physics, both from Carnegie-Mellon University. As a consulting engineer in the Architecture and Advanced Development Group, he works on aspects of FDDI technology. Currently an FDDI architect, Jerry was involved in optics development, physical layer and MAC protocols, connection management, and standards. Previously, Jerry designed digital and analog circuitry and worked on Ethernet and computer interconnect products. He holds one patent related to the FDDI physical layer and has four patents pending.



Robert C. Kochem Robert Kochem is an engineering manager in the LAN Interconnect Engineering Group. He was responsible for managing the DECbridge 500 development effort and is presently working on additional LAN interconnect products. In earlier work, Bob led the hardware team that developed the test bed for the FDDI chip set. Before joining Digital in 1986, he was employed by Codex and American Science and Engineering in management positions. Bob holds B.S.E.E. (1972) and M.E.E.E. (1973) degrees from Rensselaer Polytechnic Institute.



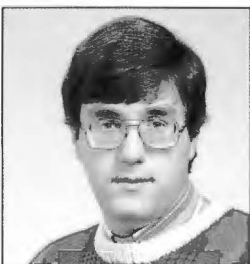
G. Paul Koning As a consulting engineer in the Distributed Systems Architecture Group, Paul Koning is involved in work related to FDDI architecture, including the specification of Digital's internal architecture and participation in the ANSI FDDI standards development. In this last capacity, he is a representative to the ANSI X3T9.5 standard committee. Since joining Digital in 1978, Paul has also worked on RSTS/E and DECnet/E development projects. He has several patents pending on various aspects of LAN and WAN technology. Paul received a B.S. (1975) in physics from Lawrence University.



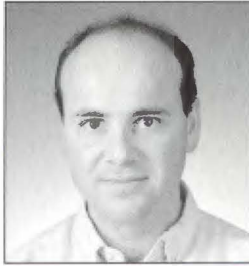
James E. Kuenzel As manager of the VAXcluster Systems Engineering consultant team, James Kuenzel is responsible for ensuring that new products and technological directions are well integrated with the existing VAXcluster system. He is currently defining the long-term strategy for the next generation VAXcluster interconnect. Prior to this, Jim was involved in the planning and development of Digital's FDDI product strategy and was a representative to the ANSI FDDI standards committee. Jim is a 1972 graduate of Philco-Ford Technical Institute.



Brian T. Mayo Brian Mayo is a principal hardware engineer working in the LAN Interconnect Engineering Group. He was the project engineer for the DECbridge 500 FDDI-to-Ethernet bridge. Previous work includes advanced development efforts in bridging other heterogeneous networks. Before joining Digital, Brian was a product development manager at Gould-Modicon in the Distributed Networking Group. He has also been both a development manager and an operations manager for Teradyne in the Systems Test Division. Brian received his B.S.E.E. (1978) from Cornell University.



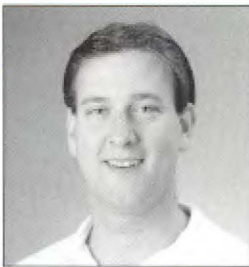
Lawrence G. Palmer Larry Palmer is a principal software engineer with the Open Systems Networking Group. He currently leads the MicroKernel advanced development project and has been a member of the ULTRIX team since joining Digital in 1984. Larry is one of the three software developers who initiated the PMAX software project for the DECstation 3100 product by porting the ULTRIX operating system to the MIPS architecture. He received a B.S. (1982) in chemistry with highest honors from the University of Oklahoma and is a member of Phi Beta Kappa.



David S. Sawyer In his position as software principal engineer, David Sawyer was the project leader for software development on the DEC FDDIcontroller 700 adapter, common node software, and system software on the FDDI tester. Before joining the Communications Systems Engineering Group, he led efforts for the design verification of the ELM chip, the FDDI physical layer protocol analysis, and the FDDI chip implementation. Dave came to Digital in 1983 after receiving a B.S.E.E. (magna cum laude) from Northeastern University. He is a member of Phi Kappa Phi, Tau Beta Pi, and Eta Kappa Nu.



Ursula Sinkewicz Currently a principal engineer, Ursula Sinkewicz has spent six of her eight years at Digital working on the ULTRIX operating system and communications. She was the project leader for networking changes in ULTRIX symmetrical multiprocessing and worked on the 2780/3780 RJE terminal emulation for the ULTRIX operating system. Ursula holds a B.S. in physics and mathematics from George Washington University and an M.S. in mathematics from Rensselaer Polytechnic Institute. Her master's thesis was a mathematical model for predicting plate subduction near earthquakes.



Craig Smelser Craig Smelser is a principal engineer who currently manages the ULTRIX Network Engineering Group. Prior to this, he worked in the Telecommunications and Network Group where he designed and developed the reusable operating system used throughout the FDDI product set. Craig also led the team that delivered the 3270 terminal option for the DECserver hardware. He earned B.S. degrees (1980) in both mathematics and computer science from Geneva College and an M.S. (1987) in software engineering from the Wang Institute of Graduate Studies. Craig joined Digital in 1987.



Barry A. Spinney Since joining Digital in 1985, Barry Spinney has been responsible for managing the development of FDDI chips, including the MAC, the RMC, and the CAM. He is currently a consulting engineer in the Distributed Systems Advanced Development Group working on next-generation LAN products. In prior work, Barry was a senior software engineer at Softech, Inc. He holds a B.S. (1979, double honors) in mathematics and computer science from the University of Waterloo and an M.S.C.S. (1981) from the University of Toronto. Barry is a member of IEEE and ACM and has four patents pending.



Bruce E. Sweet Bruce Sweet is a software engineering supervisor responsible for the development of network management software for Digital's LAN Bridge and FDDI networking products. In previous work, he was the project leader for the RBMS V2.0 product and for the LAN traffic monitor V1.0 listener software. As an individual contributor to the DECrouter 200 product, Bruce implemented the Ethernet data link and DDCCMP protocols and real-time kernel and modem control. He joined Digital in 1983 after receiving a B.S.C.S. from Northeastern University.



Fred L. Templin Fred Templin joined Digital in 1986 and has been a member of the ULTRIX Networks and Communications team until his recent transfer. He was technical project leader for the ULTRIX Ethernet adapter device drivers and for the ULTRIX FDDI engineering effort. Currently, Fred works as a sales support consultant for ULTRIX/Open Systems and networking in the Santa Clara Commercial District Sales Office. He received a B.S.C.S. (1983, University Scholar) and an M.S.C.S. (1986), both from Pennsylvania State University.



Bruce W. Thompson Bruce Thompson is a consulting engineer in the Communications Systems Engineering Group. Since joining Digital in 1985, he has worked on Digital's FDDI chip set and FDDIcontroller 700 adapter for the DECstation 5000 Model 200 workstation. Bruce also helped to define Digital's FDDI wiring concentrator architecture. Previously, he was a principal engineer at Wang Laboratories and Network Switching Systems. Bruce holds a B.S. in electrical and computer engineering from the University of Massachusetts. He has one patent on an aspect of the FDDI physical layer implementation.



William J. Tiffany As a principal engineer in the Network Systems Engineering Group, Bill Tiffany worked on the DECconcentrator 500 project and is currently involved in developing next-generation FDDI products. Earlier, he worked as a diagnostics/hardware engineer on the DECpeater 200 product. Before coming to Digital in 1986, Bill was employed by Integrity Communications, Raytheon Company, and General Dynamics Corporation. He holds a B.S. (1977) in engineering science from the University of Texas and an M.S.E.E. from the California Institute of Technology.



Stephen Towning Principal Engineer Stephen Towning is a member of the Corporate Backbone Systems Group located in Ireland. He has worked in the area of diagnostics and test for 15 years. Steve joined Digital's FDDI program in 1986 as an independent consultant to establish test and verification strategy. He became a full-time Digital employee in 1989. Steve received a B.A. (1975) with joint honors in biology and psychology from Keele University in England. He is a member of ACM and IEEE.



Henry S. Yang As a consulting engineer for the Distributed Systems Architecture Group, Henry Yang's responsibilities focus on the FDDI and Ethernet architectures. Since joining Digital in 1976, he has contributed to the development of the Ethernet adapters, synchronous communication adapters, the LAN address ROM system, Ethernet chips, the FDDI technology and products, and several Digital standards. Henry has 11 patents either issued or pending on Ethernet and FDDI. He holds a B.S.E.E. (1976, honors) from the University of Toronto and an M.S.E.E. (1988) from Northeastern University.

Foreword



Mark F. Kempf
*Senior Consulting Engineer,
FDDI Program Technical Director*

In the early 1980s, Digital introduced local area network (LAN) products based on industry-standard, 10-megabit/second Ethernet technology. These products allowed minicomputers, terminal servers, and other network devices to be connected with ease and offered unprecedented bandwidth. As local networks grew in size and began to strain the capabilities of a single LAN in the mid 1980s, Digital introduced products based on the extended LAN concept. By connecting multiple LANs with filtering and forwarding bridges, a much larger number of stations could be interconnected with greater aggregate bandwidth.

A few years after the introduction of extended LANs, it became apparent that once again the capabilities of existing LANs were being strained, but this time in two dimensions. Not only was there need for greater bandwidth in the "backbone" of the network, but with the advent of much faster workstations and servers, there was need to bring more than 10 megabits/second to a single station. The current and projected performance of workstations indicated that any new generation of LAN suitable for future Digital products would have to offer an order of magnitude increase in bandwidth delivered to a single point. It was also clear that it would have to adhere to a widely accepted industry standard, since users had come to expect the ability to interconnect equipment from many vendors.

After considerable analysis, Digital selected the emerging ANSI FDDI (fiber distributed data interface) 100-megabit/second token ring standard as the basis for our next generation of LAN products. The standard met our important requirements and showed promise of becoming widely accepted

(today, of course, it is). However, because the standard was incomplete, our plans had to accommodate the implementation of specifications that would not always be known at convenient times. Further, because FDDI had not yet been implemented, we had to expect to discover errors in the existing document and to work with the standards organization to correct them. The first group of articles in this issue of the *Digital Technical Journal* describes how we implemented the standard and some of the techniques we applied to deal with change as the standard matured.

Of course, in Digital's business, the implementation of a LAN technology is only a means to an end. The LAN technology must be incorporated into products that provide useful services to our customers. One important product in Digital's initial FDDI offering, the DEC FDDIcontroller 700 workstation adapter, meets the need for more bandwidth delivered to a single station. But LAN users have come to expect more: manageability, interoperability, and a generally radial wiring scheme; and they have large existing LAN infrastructures. Therefore, it was essential to introduce FDDI with a set of products that addressed all these needs as well. Digital's FDDI wiring concentrator, the DECconcentrator 500, permits highly reliable, manageable, radial wiring schemes. DECelms (extended LAN management software) provides network management capability for Digital's FDDI and preexisting extended LAN products.

It is important to remember that Digital does not view FDDI as a replacement for Ethernet, but rather as its complement. The large number of existing Ethernet LANs can be connected to FDDI using another element in Digital's initial FDDI product offering, the DECbridge 500 FDDI/Ethernet bridge. This product allows multiple Ethernets to be interconnected using FDDI as a backbone. Ethernet stations can communicate directly with stations on FDDI as well as with stations on other Ethernets. The second group of articles in this issue describes how we developed these FDDI products.

During the entire development process, we relied on several design strategies. One emphasized the importance of designing the entire system that comprises the product set, not just individual products. Another was the decomposition of problems. Early in the design process, we identified functions that appeared to be independent of others. We took into account how each component that implemented these functions would be used in the

various products and how the products would interact with each other. In our development plans, we assumed that once a particular independent component was designed, implemented, and tested, it could be depended on to work as expected when other components were added around it. Naturally, we could not assume that our initial assessment of independence was infallible, and we remained alert for unexpected interactions.

Another related design strategy was partitioning to reduce risk. As mentioned before, the FDDI standard was evolving and subject to change. Therefore, we physically partitioned our design to closely mirror the separate sections of the standards document. This approach helped limit to a single component the impact of standards changes.

The final important design strategy can be summarized as "analyze everything you can, simulate what you can't analyze, and prototype what you can't simulate." The work, expense, and recovery time from errors increases with each step through this progression, and the advantages are obvious. Of course, it is also obvious that this summary is simplistic since it is impossible to analyze, simulate, and prototype exhaustively. It is necessary to step into the gray area of risk assessment and engineering judgment to make satisfactory progress.

FDDI LAN technology development and product development was done in three coordinated and somewhat overlapping phases. In the first phase, we concentrated on analysis and simulation of the underlying FDDI algorithms specified by the standard. We wanted to ensure that they actually described mechanisms that would produce a reliable, high-performance LAN. During this phase we also implemented the standard in silicon and software, and combined these components to form complete FDDI test stations.

Since standards documents, like all written documents, are subject to interpretation, two separate teams were given the task of implementation and verification. We used the test stations, first in a simulation environment, to execute test scripts developed directly from the standards documents to verify standards compliance. The scripts were also used to verify details specific to our implementation and had the additional benefit of making regression testing after incremental changes relatively easy.

In the second phase we concentrated on proving that significant numbers of FDDI stations, which had previously been shown to work individually, could

be interconnected to form large reliable rings. This was the most important application of the heavily instrumented test systems. Although analysis and simulation can and did yield important results in this area, the complex interactions between large numbers of asynchronous stations overwhelm both analytical techniques and available computing power.

In the third phase, the focus was on producing products. From a logic design and software standpoint, Digital's FDDI products are largely derived from the test systems used in the previous phases. Of course, laboratory test capabilities were removed and major changes in power and packaging were made; but this approach significantly reduced both the opportunity for introducing new errors and the time to market. We were also able to use the capabilities of the laboratory versions to help verify the correct function of the products.

Numerous other activities that contributed to the effectiveness and timely delivery of the products were carried out simultaneously with engineering design. For instance, Digital maintained a significant presence at the FDDI standards committee to apprise the committee of various technical problems we found in the standard, to offer solutions, and to ensure that our implementation reflected the intent of the standard. In addition, the close working relationships fostered between various organizations, especially between development and manufacturing, resulted in products with a good balance between time to market, function, performance, and manufacturing cost.

The articles in this issue of the *Digital Technical Journal* go into much greater detail on the subjects I have touched on. I hope you find them interesting and informative.

Fiber Distributed Data Interface Overview

After exploring various alternatives to second-generation local area networks (LANs), Digital selected the fiber distributed data interface (FDDI) system. FDDI implements the International Standards Organization (ISO) physical layer and the media access control sublayer of the data link layer. This system is based on a 100-megabit-per-second fiber-optic ring network and uses a timed-token protocol to coordinate station access to the network. Digital has developed the FDDI base technology, including very large-scale integration (VLSI) chips and software. These chips, licensed to Advanced Micro Devices and Motorola, Inc., provide high-quality alternatives in the market and foster cost reduction. Digital's implementation of FDDI, including backbones in extended LANs, as well as high-speed interconnection of workstations, servers, and central computers, makes available a complete range of system products.

As the use of local area networks (LANs) continues to grow at an exponential rate, many large networks with Ethernet backbones are reaching their usable capacity. In addition, the explosion in the use of high-performance workstations is placing increasing demands on network performance as larger volumes of data pass from station to station. Several years ago, Digital recognized this growth trend and began to plan and develop a second-generation LAN that would follow Ethernet and provide an evolutionary path to higher performance. The selection of FDDI as the second-generation LAN was made with great deliberation. This paper explores the criteria for that choice and the history of the FDDI system to the present. The theory of FDDI operation, the development of the FDDI technology's role in Digital's networks, and the resulting products are also presented and discussed.

Selection of FDDI

Many of the same criteria originally used to select Ethernet were again used to evaluate the application environment for the new LAN. The need to consider migration from the popular LANs currently in use presented the only new concern. Paramount among the reasons for selecting the FDDI technology were its tenfold increase in bandwidth over Ethernet, its consistency with other IEEE 802 LANs, and the standardization effort already begun in the American National Standards Institute (ANSI).

It is important when developing a new LAN technology to be sure the differentiation from current capabilities is sufficient to warrant the necessary investment. Moreover, a new LAN is a significant investment for a customer and should offer a large increase in capabilities such as speed and throughput. Without this increase, the technology will have a short life span (a few years) and technology such as parallel use of existing LANs to double capacity will be a realistic alternative to a wholesale replacement of the LAN. However, it is important not to take such a large technological step that exotic and complex implementation constraints become necessary. A LAN that does not lend itself to a very large-scale integration (VLSI) logic solution will not integrate well in the computer interconnect environment and will not be cost effective for wide-scale use.

FDDI, with its tenfold increase in speed, provides significant differentiation from Ethernet/802.3 and current token ring and bus technology to justify the new investment. Examination of the clocking, buffering, and state machine needs of the media access control (MAC) sublayer of the data link layer also showed that the FDDI technology could be implemented in several VLSI chip components. Further, as silicon technology improves, cost reduction is possible, enhancing the longevity of the FDDI LAN technology.

Migration is another important factor in the selection of a new LAN. Many devices exist with

embedded LAN interfaces that will never be directly connected to any other LAN. Still other devices will not benefit from the added capabilities of a higher speed interconnect. Examples of such devices include a processor or workstation too slow to send or to receive data any faster from the network, an output-limited print server, or communications servers with other, more constraining, I/O ports. It would never prove cost effective to upgrade these devices to a new higher speed LAN interface but, as a group, they would need to obtain unconstrained (no bottleneck) connectivity to the services of the new LAN for smooth migration allowing protection of the investment in devices and LANs already in place.

Standards are important for networks as a way to ensure consistent interface compatibility and interoperability for communications services. As the IEEE 802.1d standard readily demonstrates by attempting to interconnect dissimilar LANs at the MAC sublayer, some standards are more compatible than others. A common logical link control (LLC) format or the format within the MAC frame allows a smooth migration between LANs by allowing a transparent bridge to provide protocol-independent translation between LANs. Ethernet, the forerunner of IEEE 802.3, does not use the IEEE 802.2 formatted LLC and, therefore, migration of those frames is more challenging.

Lastly, the media selected for the new LAN has to be consistent with current and projected future needs. Even for slower speed LANs, fiber-optic media is gaining in popularity because of its superior qualities in spanning greater distance, its noise immunity, and its declining user cost.

The FDDI technology meets the necessary selection criteria as an emerging American National Standards Institute (ANSI) standard using fiber and allowing other media in place of fiber while providing a tenfold increase in speed. Migration of some devices could be affected directly by changing controllers, and bridging between LANs could allow smooth migration of all existing devices.

FDDI History

Both the ANSI FDDI standards and the industry-wide implementations of these standards have evolved slowly. A variety of factors have contributed to this course of development. The FDDI ring was originally invented at Sperry and Burroughs Corporation. The ring was to be used as a machine room interconnect between processors and storage systems, much like the Computer Interconnect components are used in

Digital's VAXcluster systems.¹ The timed-token, media access control protocol itself was first publicized in 1982 by Bob Grow while he was at Burroughs.²

As a machine room interconnect between processors and storage systems, the initial ANSI standard requirements on the FDDI technology were quite different from today's needs. In particular, as a machine room network, the number of stations was assumed to be relatively small compared to a LAN and unstructured cabling was to be used. Since most machines were always running, fault recovery could be accomplished by having a dual ring with failover to the secondary ring. Thus, a failed station or cable could be isolated without partitioning the ring. The FDDI technology retains this property today. However, that basic operation capability is insufficient in a LAN environment with structured cabling requirements and a large number of stations, any number of which might be unplugged or turned off by users. Therefore, we have expanded the definition of the FDDI technology to include such products as concentrators and adapters.

In 1982 the FDDI technology was brought to the attention of the ANSI X3T9 committee, which develops standards for I/O interconnects and channels. Since FDDI was intended to be used as a machine room interconnect, this committee was the appropriate arena for study. Over time, however, as the need for a 100-megabit-per-second LAN emerged, the FDDI technology evolved into a local area network. Some classic standards territory conflicts developed between IEEE 802, the group that defines all the LAN standards, and this ANSI committee.

While FDDI was evolving from a machine room interconnect into a general-purpose LAN, the requirements changed. For a machine room interconnect, some management operation to install and initialize the network might reasonably be allowed. For example, the manager might set the values of various parameters to control the operation and performance of the interconnect network. However, in a general-purpose LAN, manager involvement is unacceptable. For simplicity, robustness, and ease of management, the industry widely accepts that LANs must autoconfigure, also called "plug-and-play." Inevitably, FDDI was required to exhibit the attributes of a true local area network. Since the FDDI technology and standards were already in development when this evolution of requirements occurred, the ANSI committee made an attempt to accommodate the following two views of the network: first, the network should be completely configurable with almost every parameter and policy

controlled by management; and second, the network should be a local area network with the corresponding attributes of simplicity and autoconfiguration. Incorporating both models into the ANSI FDDI standards made the standards complex and was one factor contributing to the eight-year-long time period to completion.

Theory of Operation

FDDI stations are composed of the basic elements defined by the FDDI standards. The physical medium dependent (PMD) layer specifies the fiber-optic interface and data driver and receiver operation for FDDI stations.³ The physical layer protocol (PHY) specifies the encoding and framing of data and control information exchanged between stations.⁴ The control information exchanged varies with the physical layer protocol type, which is either PHY-A, PHY-B, PHY-M, or PHY-S. The MAC sublayer specifies the protocols for logical ring formation and control, for the transmission and reception of packets at a station, and for the repetition and stripping of packets on the ring.^{5,6} Station management (SMT) provides *n*-layer management and a local manage-

ment interface to the PMD, PHY, and MAC layers.⁷ Together these components support an IEEE 802.2-compatible logical link control capable of supporting client protocols such as the Digital networking (DECnet) protocol, open systems interconnection (OSI), local area transport (LAT), and the transmission control protocol/internet protocol (TCP/IP). Stations utilizing the FDDI components can take several forms such as single attachment stations (SASS), dual attachment stations (DASS), and dual attachment concentrators (DACs). An architectural model is shown in Figure 1.

Configurations of FDDI Components

A single attachment station is the simplest configuration and consists of the fundamental FDDI components arranged as shown in Figure 2. There is a single incoming data path and a single outgoing data path with a MAC in between.

Dual attachment stations, as shown in Figure 3, include a second physical layer and provide connections to a secondary ring for use in the event of breakage on the primary ring. Under fault-free operating conditions represented by the THRU STATE

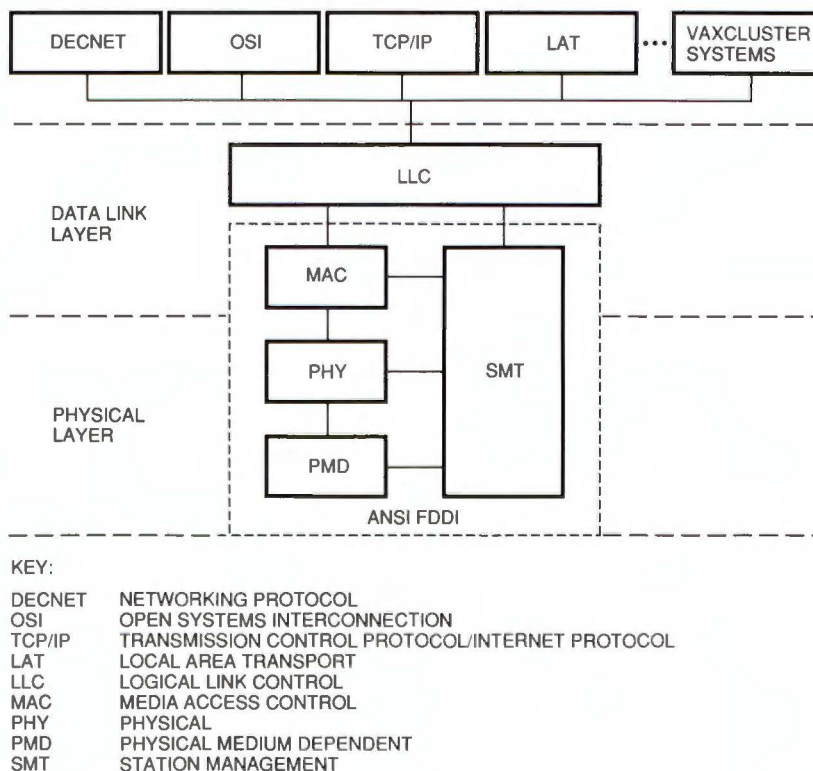


Figure 1 Architectural Model

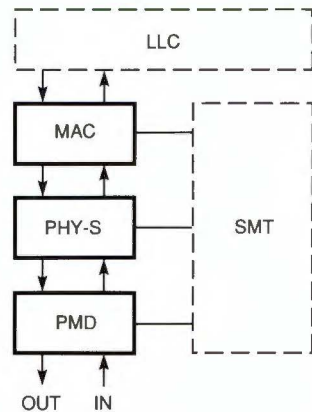


Figure 2 Single Attachment Station

area of Figure 3, the primary data path enters through PHY-A, travels through the MAC, and exits through PHY-B. The secondary data path enters through PHY-B and exits directly through PHY-A. If a discontinuity is detected in the primary data path, either within the station or on one of the PHYs, the station wraps the two data paths, thereby providing an alternate route through the secondary data path. This situation is shown in the WRAP A STATE area of Figure 3.

A dual attachment concentrator builds on the dual attachment station by adding additional master PHYs (PHY-M) in the primary data path as shown in Figure 4. Single attachment stations can then be included in the ring by connecting them to the additional PHY-Ms in the concentrator.

An FDDI LAN is formed by joining multiple stations to form a logical ring topology. The logical ring can take two physical forms, a dual trunk ring and a tree ring.

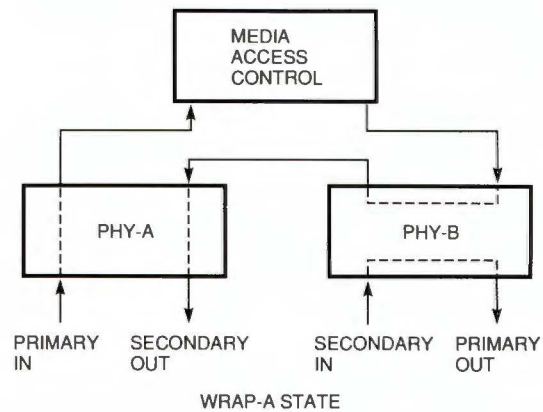
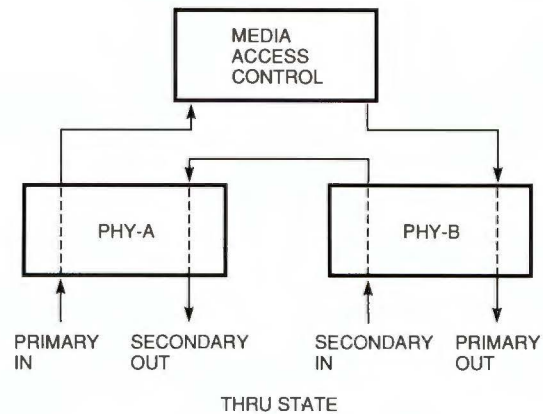


Figure 3 Dual Attachment Station States

As depicted in the upper portion of Figure 5, the dual trunk ring is formed by connecting dual attachment stations and concentrators to form a LAN. This portion of the LAN consists of two data paths in opposite directions, called the primary

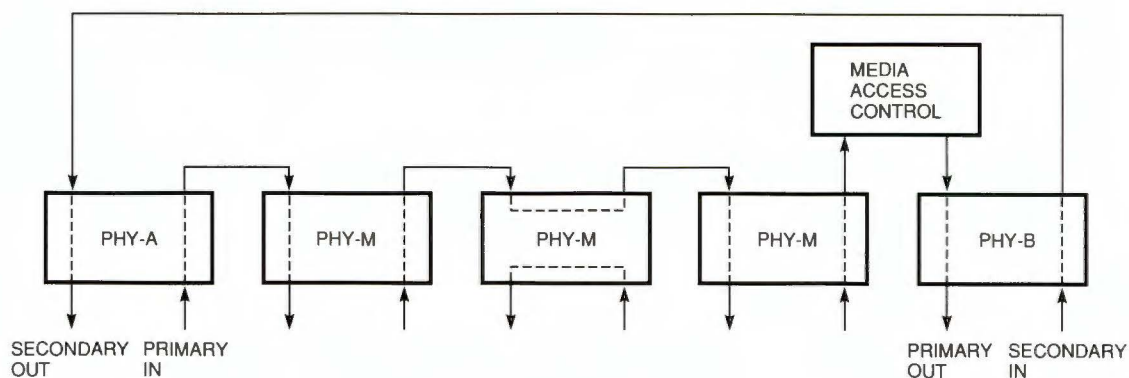


Figure 4 Dual Attachment Concentrator

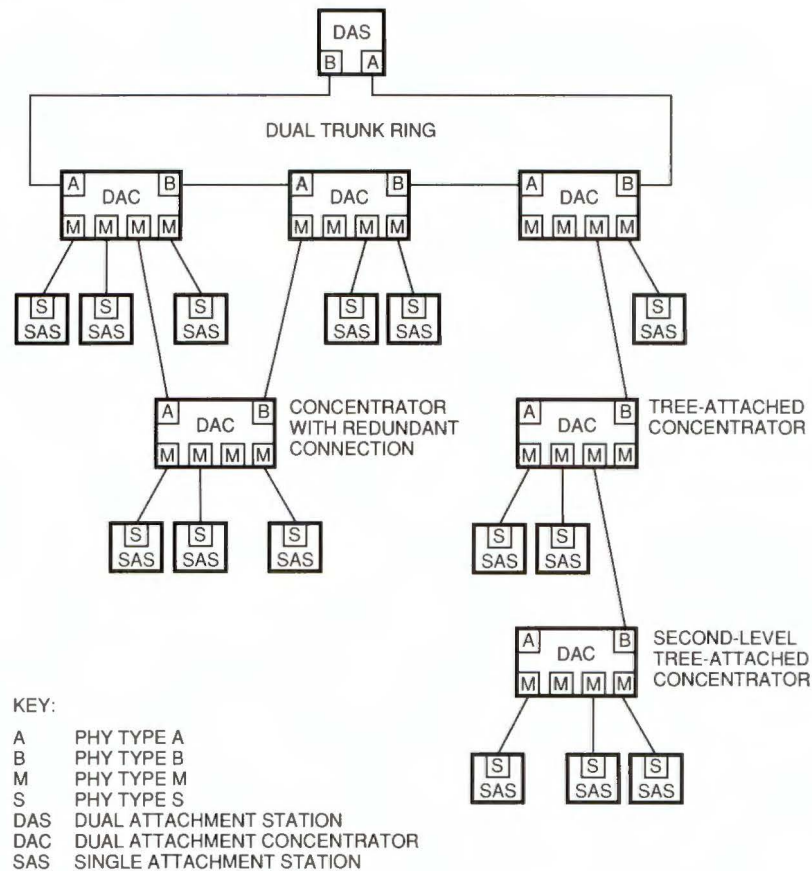


Figure 5 FDDI Dual Ring of Trees Topology

and secondary rings. Under normal operation, data flows on the primary ring from station to station. In the event of a cable or a station breakage, the stations adjacent to the fault join the primary and secondary rings and then use the secondary path to reestablish a logical ring.

A tree ring can be formed by connecting stations or concentrators to the PHY-Ms of a concentrator as shown in Figure 5. In this formation, the primary data path descends down each branch of the tree passing through each station in the tree, until it finally reemerges into the dual trunk ring.

Media Access Control Sublayer Operation

As mentioned previously, the MAC sublayer provides the protocols for logical ring formation and data packet operations. To initialize the ring, all MACs first enter the claim process to determine which MAC will generate the token and to establish the token rotation time for the ring. Each station continuously transmits claim frames that contain the

station's requested token rotation time. When a MAC receives claim frames with times shorter than its own, or equal to its own but from a station with a numerically larger address, it yields, stops sending claims, and repeats the claims received from its neighboring station. Eventually the station with the shortest or "winning" time will receive its own claim. This station then generates the token and the ring enters the operational state.

If the claim process does not complete within approximately 100 milliseconds, the MACs in the ring perform a beacon process to confirm continuity of the ring. Special beacon frames are transmitted continuously by all MACs until a beacon is received, at which point it stops transmitting. This process continues until one MAC transmits and receives its own beacon, indicating ring continuity.

To transmit data packets, the MAC first waits for the token to arrive, holds it, and, then, transmits the packets, reissuing a new token at the end of the transmitted packet stream. The time allowance

a station has to transmit packets after receiving a token is equal to the token rotation time established by the claim process.

Packets received by a MAC are either repeated for reception by the next station or stripped from the ring. In addition, a MAC may store a copy of a packet for use by the station. After transmitting a frame, a MAC is responsible for stripping that frame from the ring after the frame makes exactly one traversal of the ring. Frames left unstripped are considered no-owner frames and can circulate the ring forever. This condition floods the station to which the frame is addressed and is thus detrimental to ring performance. A MAC typically strips frames by comparing the source address in the frame with the MAC's own address. The MAC strips any frame it has previously sent but repeats the frame, otherwise.

FDDI Ring Formation

An FDDI ring is formed in several stages, beginning with the successful establishment of point-to-point links between all adjacent PHYs. These link connections are made by the connection management protocol (CMT).⁷ This protocol defines control signal exchanges to synchronize the two ends of the link, to exchange information about the PHY type (i.e., A, B, M, S) and link testing requirements of each end to perform link quality testing, and to finalize the connection for normal operation. Before the link establishment enters its final phase, the PHY types of the two ends of the connection are compared, and the connection is allowed if the end types conform to specific connection rules. These rules are carefully established to ensure that rings are configured correctly and to prevent miscabling, which could cause partitioned and unnecessarily wrapped rings. An established link is continually monitored for errors indicated by the reception of improperly encoded data and is shut down if those errors exceed a predetermined threshold.⁸

After the PHY connection is fully established, the station's MAC is inserted into the ring, and the claim process begins. Digital's stations all use a default requested token rotation time of eight milliseconds in their claim process to ensure a token rotation time, and hence traffic latency, similar to that experienced in other LANs like Ethernet. FDDI provides high aggregate bandwidth and, thus, delivers the low delay essential to many LAN applications such as disk I/O, interactive graphics, and remote procedure calls. Higher settings of the requested token rotation time result in

very large delays on the ring while improving the efficiency only slightly.⁹ Therefore, after extensive performance modeling, Digital decided to use this default value for the requested token rotation time. The resulting network operates with low delay and high bandwidth as the default and does not need complicated network management procedures to achieve this level of operation.

Several unique but harmful conditions in FDDI stations must be addressed. Among them are the prevention of and protection against duplicate addresses, no-owner frames, and the stripping of frames sent by bridges or end stations that have multiple addresses.¹⁰ Because several of the algorithms fundamental to the operation of the FDDI technology use the stations' addresses, the presence of two stations with the same address causes numerous malfunctions, ranging from beacon-claim oscillations to blocked communications between stations. Frames not properly stripped from the ring can circulate forever, flooding the stations that copy these frames. To protect against strip errors, Digital's chip set has several built-in mechanisms. Digital also greatly improved the data integrity of the ring. This improvement is particularly important to token ring architecture where messages traverse virtually all of the links in the network before arriving at their destinations. Robustness in the face of link bit errors becomes extremely important. Digital designed several improvements to the basic FDDI algorithms, and the ANSI committee adopted them to improve the undetected error rates on a network.¹¹

Role of FDDI in Digital's Networks

The FDDI technology is more than just another data link interface that allows the use of transmit and receive fibers between devices. Digital's decision to embark on the FDDI development effort was a major program undertaking involving the development of VLSI chips and, subsequently, FDDI software and hardware products. Although the development of chips may seem to be at the heart of the FDDI program, chips are certainly not the products that help customers solve problems. Chips comprise only a small portion of each large printed circuit board, but this portion is an important one.

Before focusing on chip development, Digital carried out a large simulation effort to ensure the ANSI standards were correct and complete. Once the standards were verified, modeling was performed to produce chips that met these standards. Allowances and trade-offs were made for unfinished

sections and future standards migration. Real products were then planned around the chips as the use of FDDI was threaded into Digital's network architectures and existing products. An implementation strategy for each product was then formulated to balance the risks, the resources, and the timeliness of customer needs.

It was important for Digital to understand migration of current products and to think forward to the needs beyond the initial program goals. LAN management and host connections take the FDDI technology beyond a simple high-speed backbone for the extension of bandwidth-limited existing LANs. Because Digital considered the FDDI technology beyond its use as just another new data link, this technology is the natural choice as the next step in network evolution.

At the onset of Digital's FDDI program, and at several points during its development, a number of key program-wide decisions and policies were adopted. The commitment to simulation, both in breadth (e.g., chip design, board design, and software) and extent (e.g., behavioral models, gate-level models, integration of operational software with simulation models) was essential to the success of the FDDI technology development. All the chips developed were fully functional in their first pass in silicon, and the integration of the controlling and test software with the chips was fast and smooth.

Given the extensive investment in ensuring correctness of the FDDI technology, all FDDI products were to use the same FDDI building blocks, including the chips and controlling software.¹² Such sharing and reuse of investment eliminated the duplication of effort, guaranteed the consistent operation of all products, and shortened the time to market. To ensure an even broader use of this technology, major portions were made available on the open market.

Another key program decision was the adoption of electrically alterable read-only memory (EAROM) in place of traditional read-only memory (ROM) to store the firmware in each product. SMT, which is implemented almost entirely in firmware, was a rapidly evolving specification while the products were under development. To accommodate these ongoing changes and the threat of change after product shipment, EAROM was included in all products to allow firmware to be updated remotely over the network. As a result, EAROM reduced the cost of product enhancement by eliminating the need to change ROM in the field or swap out boards and rework them at the factory.

Deployment of FDDI

As a baseline effort in deploying FDDI in products, Digital developed the FDDI design corner shown in Figure 6, consisting of chips and FDDI control firmware for use in all applications of FDDI. In addition to using the design corner in all of Digital's products, two of the chips critical to FDDI interoperability, the MAC and the ELM, together with the SMT firmware, which controls the chips and the behavior of the station, were licensed to Advanced Micro Devices and to Motorola, Inc. for manufacture and sale on the open market. Widespread availability of this technology will foster competition to drive down cost and increase levels of interoperability and consistency among FDDI implementations.

FDDI-to-Ethernet Bridge

The Ethernet/802.3 bridge to the FDDI network is an important device in the first product set offered for FDDI connectivity. The DECbridge 500 product provides the smooth migration of all Ethernet and IEEE 802.3 devices currently in the marketplace to the increased backbone bandwidth of the FDDI technology. Most local networks are LAN-based

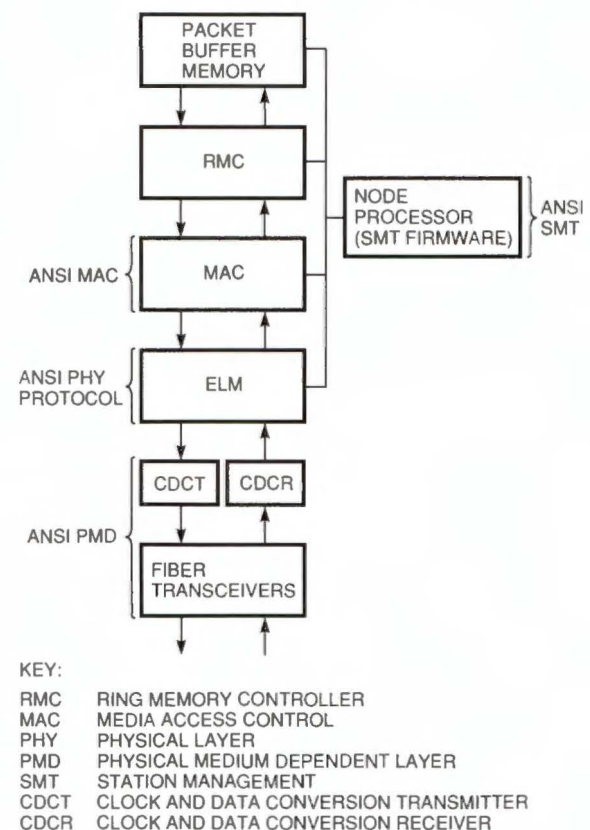


Figure 6 FDDI Design Corner

with the LANs being interconnected to form extended LANs by the use of bridges. The bridges are transparent to all users' protocols on the LANs. This capability for transparent interconnection is one of the keys to the instant use of and easy migration to the added bandwidth and other benefits of FDDI. However, providing the transparent interconnect is difficult. Frame formats, bit ordering, padding fields, and even the length of the packets differ between the Ethernet and IEEE 802.3 LANs. These differences force the frames to be translated as they pass through the DECbridge 500 device. This translation, coupled with a stiff requirement for data integrity for the MAC bridging standard IEEE 802.1d and the performance needs of dealing efficiently with packet rates approaching 500,000 per second, created a formidable design task.¹³

FDDI Concentrator

To make the FDDI technology widely acceptable as the next generation LAN, several of the logistical obstacles inherent in a ring topology had to be overcome. Connecting new stations to a LAN can be frequent events. These events must not be restricted or cause a disruption if the LAN is to be truly usable. The nature of a ring topology causes the network to break whenever a station is removed or while a station is being inserted. The dual ring accommodates one such event by wrapping at the adjacent stations, but two or more such events are disastrous because they partition the network into isolated parts. A concentrator can solve this problem by connecting to a station, testing the link between the station and the concentrator, and then splicing the new station into the ring with minimal disruption. At the same time, a concentrator can automatically drop a station out of the ring should the station malfunction or lose power.

In addition to the obstacle of connecting new stations, it is inconvenient to map a ring topology into star wiring, which is well known and widely used for its convenience and manageability. Concentrators can turn a logical ring into a hierarchical star-wired network, thus solving this mapping problem.

Digital's DECconcentrator 500 product was built to support simple installation and network configuration, existing building cabling plants, and the cost savings and simplicity of single attachment stations. This FDDI concentrator is low cost and flexible, with a dual attachment and up to eight master ports for connecting single attachment stations or additional concentrators into the ring. With the concentrator, any of the described FDDI topologies

can be created to best suit a customer's cabling, network management, and availability needs.¹⁴

FDDI for Workstations

Digital designed a simple high-performance FDDI adapter for TURBOchannel workstations and servers in conjunction with the design of the DECstation 5000 reduced instruction set computer (RISC)-based system.¹⁵ The aim of this product was to produce an FDDI interface that was both inexpensive and high performance. With this adapter and ULTRIX operating system support, Digital attempted to address many of the shortcomings that have plagued most adapters and operating systems attempting to achieve high performance.¹⁶

The applications of the FDDI technology for workstations and servers depend on the particular choice of user applications and on the network configuration. One application is the use of the FDDI technology for a work group of high-performance workstations and servers. Some high-performance I/O-intensive applications need more bandwidth than a large, shared Ethernet can handle. As a result, a work group may form around an FDDI network. One application example is called visualization, where a large volume of real-time graphics or imaging information is transmitted over the network from a compute server. Various applications of digital image transmission have few real-time latency requirements, but still require a large bandwidth.¹⁷

Another application of the FDDI technology in workstation environments is to achieve higher bandwidth for server/server and backbone interconnection. A shared Ethernet for client/server as well as server/server communications can be overloaded, particularly if the population of servers or workstations is large. FDDI can be used as a backbone to provide higher bandwidth between the servers. Alternatively, the network may be a hybrid, where the FDDI network is used for high client/server bandwidth by some systems and for high server/server bandwidth by others. Hybrid networks often have multiple FDDI networks and multiple Ethernet networks. This configuration is typical of evolving extended LANs, as new technology such as FDDI is added into the existing network infrastructure.

FDDI in the Future

As a new technology, FDDI has a long future with opportunities for greater levels of circuit integration, lower cost designs, and alternative media types to match the wide range of possible applications. Significant progress has already been made

in this direction with the introduction of the DECcontroller 500 adapter, the most compact, least expensive, and one of the highest performing workstation adapters in the industry today. In addition, the ground-breaking technical work and standards committee activity regarding various types of copper media offers tremendous opportunities for cost reduction which will enable broader utilization of the performance offered by the FDDI technology.¹⁸

In addition to the original physical medium dependent standard specified by ANSI for FDDI, an additional standard has been developed for operation on single-mode fiber over distances up to 30 kilometers (km). This standard will be particularly useful in deploying an FDDI network that must span one or more distances greater than 2 km such as a campus backbone. Other medium types are currently under investigation to provide optimization in areas such as cost and ease of installation.

Acknowledgments

We would like to thank the Telecommunications and Networking LAN Architecture and Development teams. Without their hard work none of the many technical advances made in bringing the FDDI technology to market would have been possible. We would also like to thank Bob Krueger for his role in successfully leading the entire program. Finally, we would like to thank Mark Kempf, Paul Koning, and Tony Lauck for their timely review of this paper.

References

1. N. Kronenberg, H. Levy, W. Strecker, and R. Merewood, "The VAXcluster Concept: An Overview of a Distributed System," *Digital Technical Journal*, vol. 1, no. 5 (September 1987): 7-21.
2. R. Grow, "A Timed Token Protocol for Local Area Networks," *Proceedings of IEEE Electro/82 Conference*, Boston, MA (May 25-27, 1982).
3. *Token Ring Physical Layer, Medium Dependent (PMD)*, (International Standards Organization, reference no. ISO 9314-3, 1990).
4. *Token Ring Physical Layer Protocol*, (International Standards Organization, reference no. ISO 9314-1, 1989).
5. *Token Ring Media Access Control (MAC)*, (International Standards Organization, reference no. ISO 9314-2, 1989).
6. H. Yang, B. Spinney, and S. Towning, "FDDI Data Link Development," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 31-41.
7. *FDDI Station Management (SMT) Preliminary Draft Proposed American National Standard*, ANSI X3T9/90-X3T9.5/84-49, REV 6.2 (May 1990).
8. J. Hutchison, C. Baldwin, and B. Thompson, "Development of the FDDI Physical Layer," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 19-30.
9. R. Jain, "Performance Analysis of FDDI Token Ring Networks: Effect of Parameters and Guidelines for Setting TTRT," DEC-TR 655 (Maynard: Digital Equipment Corporation, September 1989).
10. H. Yang and K. Ramakrishnan, "Frame Content Independent Stripping for Token Rings," *Proceedings of the ACM SIGCOM '90 Symposium* (1990): 276-286.
11. R. Jain, "Error Characteristics of Fiber Distributed Data Interface (FDDI)," *IEEE Transactions on Communications*, vol. 38, no. 8 (August 1990).
12. P. Ciarfella, D. Benson, and D. Sawyer, "An Overview of the Common Node Software," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 42-52.
13. R. Kochem, J. Hiscock, and B. Mayo, "Development of the DECbridge 500 Product," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 53-63.
14. W. Tiffany, P. Koning, and J. Kuenzel, "The DECconcentrator 500 Product," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 64-75.
15. U. Sinkewicz, C. Chang, L. Palmer, C. Smelser, and F. Templin, "ULTRIX Fiber Distributed Data Interface Networking Subsystem Implementation," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 85-93.
16. W. Hawe and K. Ramakrishnan, "The Workstation on the Network: Performance Considerations for the Communications Interface," *Proceedings of the Second Workshop on Workstation Operating Systems*, Asilomar Conference Center, CA (September 1989).
17. K. Ramakrishnan and W. Hawe, "Performance of an Extended Local Area Network for Image Applications," *Proceedings of the Fifth Annual Phoenix International Conference on Computers and Communications*, Scottsdale, AZ (March 26-28, 1986).
18. S. Ginzburg, W. Mallard, and D. Newman, "FDDI over Unshielded Twisted Pairs," *IEEE Proceedings, Eighteenth Conference on Local Computer Networks* (October 1990).

Development of the FDDI Physical Layer

The engineering development of the FDDI physical layer resulted in the delivery of components, specifications, and protocols. The development presented new design problems related to the technology and to the operation of token rings. The choice of the most appropriate technologies for the chip set was based on technology issues, risk control, and costs. The chip set that emerged after the physical layer functions were partitioned uses both ECL and CMOS technology. Further, three design problems of general interest arose during development: the elasticity buffer and circuitry related to the distributed clocks in an FDDI LAN, the multimode fiber-optic link using light emitting diodes, and the media error processes as related to correctness and fault isolation.

The fiber distributed data interface (FDDI) is a multi-access, packet-switching local area network (LAN) that operates at 100 megabits (Mb) per second. The physical layer of FDDI—the topic of this paper—connects many stations, each of which may transmit information to any other station in the network. As in other LANs, packets of user data are encoded according to the physical layer protocol and are transmitted as a serial data stream over a physical media to other stations of the LAN. FDDI, however, is unique in its use of hundreds of individual, point-to-point, fiber-optic connections that form a ring network in the physical layer. The resulting LAN offers both a high data rate and a total physical extent of up to 100 kilometers (km).

The development of physical layer hardware, used in all FDDI products, included the physical protocol (encoding/decoding) device, a receive clock recovery device, a local clock generator, and optical transmitters and receivers. This paper focuses on development of the physical layer hardware and describes some aspects of the design in detail. We first review the operation of the physical layer and the functional partitioning of the implementation. We then present detailed discussions of the distributed clock scheme, the design of an optical link, and the methods to control the effects of bit errors in the physical layer. Some of the results of the development to improve the performance, correctness, and reliability of FDDI described here have been incorporated in the American National Standards Institute (ANSI) FDDI standards.

Operation of the Physical Layer

The FDDI physical layer is a collection of point-to-point links forming a “ring.” The operation of the layer is described in terms of physical links, physical connections, and the functions of individual stations. The many station types allowed by the ANSI FDDI standards are constructed with a simple physical layer functional block called the PHY port.

A physical link contains a transmitter, a receiver, and a segment of physical medium which conducts the bits of a packet from one station to a second station. The topology of FDDI is arranged so that the collection of physical links forms a closed path, or ring, as shown in Figure 1. This simple topology illustrates the basic concepts common to even the most complicated topologies for FDDI. Each bit of information received from one physical link is transmitted onto another physical link until the information travels around the loop and returns to where it started. The FDDI protocols provide for a single originator of data packets; other stations repeat the data so that each station on the ring receives the packet of information. The collection of many point-to-point links forms the ring, which is viewed as a multiaccess medium by the users.

The basic element in the topology of an FDDI LAN is the physical connection. A physical connection contains two physical links, also shown in Figure 1. Within the station, the circuitry that implements physical layer functionality needed for one physical connection is called the PHY port. The physical connection is a full duplex connection between

exactly two PHY ports. Neighbors in the ring directly exchange the control information for each connection, allowing control protocols in FDDI station management (SMT) to establish the shared states for a connection: in-use, starting, and disconnected. The status "in-use" indicates that a connection is part of the ring; other states indicate it is not. The control information exchanged over the physical connection is used to autoinitialize and autoconfigure the connections in the LAN, a method of operation currently unique to FDDI rings.

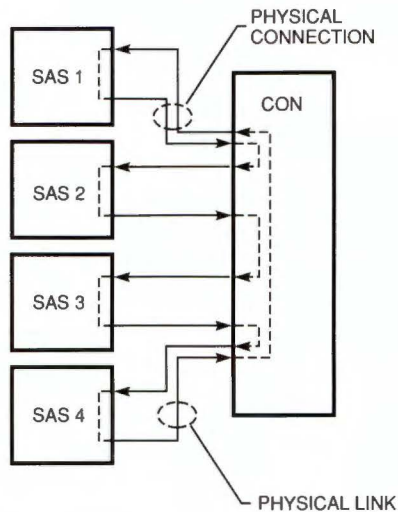


Figure 1 Physical Links Forming a Ring for the FDDI Physical Medium

There are several types of FDDI stations, and different types can support different numbers of physical connections.¹ A single attachment station (SAS) (as seen in Figure 1) can establish one physical connection with a single neighbor. The dual attachment station (not shown) has two PHY ports and may establish physical connections with two neighbors. A concentrator (CON) is a type of station that can establish connections with many neighbors, thereby providing attachment points for other stations. The CON shown in Figure 1 interconnects its PHY ports internally to configure a single ring.

Figure 2 shows the functions of and flow of data through a PHY port which implements the FDDI physical layer protocol (PHY) standard.² Data packets to be transmitted over the LAN are passed as a stream of bytes from the data link to the physical layer. Each byte contains two PHY symbols, and each symbol represents 4 bits of user data. The FDDI coding scheme, called 4B/5B encoding, translates each

symbol into a code group containing 5 code bits. This encoding limits the maximum time between transitions on the media (allowing clock information to be derived from the coded signal) and bounds the low-frequency components of the signal spectrum. The code bits are then converted to a serial stream and transmitted as optical pulses on the fiber-optic media.

The station is coupled to the media with a media interface connector (MIC). The MIC provides the concrete interface necessary for interoperability between equipment from multiple vendors. The FDDI Physical Layer Media Dependent (PMD) standard specifies mechanical and optical properties of the MIC.³ The MIC includes both a transmit and a receive interface.

Signals received from a connection are decoded by the PHY port for processing by the station. The optical input signal is translated to an electrical signal. The remote bit clock is extracted from the signal and used to recover logic levels corresponding to

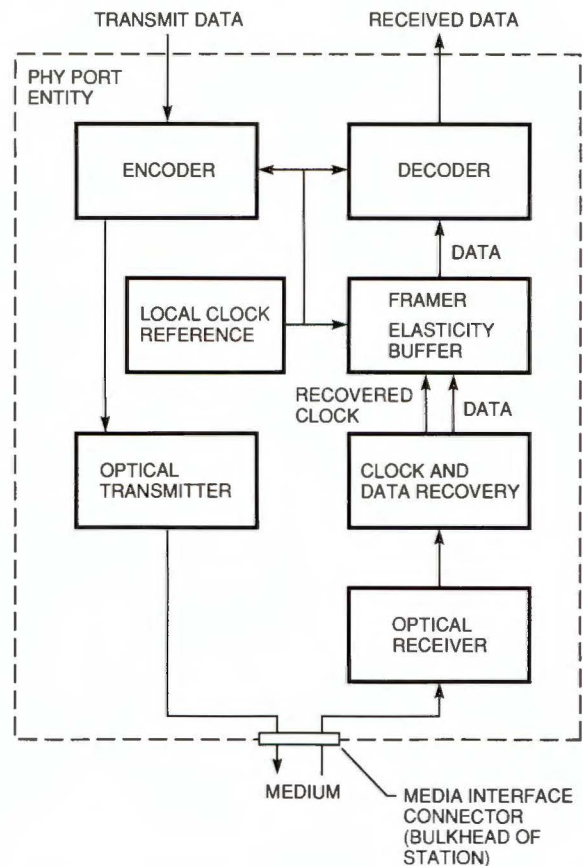


Figure 2 Functional Model of the Physical Layer for an FDDI Station

the individual bits. A framer then establishes the original code group boundaries and converts the serial code bit stream into parallel form. Also, the elasticity buffer synchronizes the received data to the local clock reference and accounts for the frequency difference between the local and remote clock references. Finally, code groups of 5 code bits are decoded into symbols, and symbols are correctly paired to form the data bytes which represent the received data. These data bytes are passed either to another PHY port or to the data link layer. A later section, Operation of the Distributed Clock Scheme, expands on the elasticity buffer design.

We have so far described the FDDI physical layer in terms of PHY ports and the physical connections between them. These basic elements form the physical layer for all types of FDDI stations. Different types of FDDI stations have one or many PHY ports, but the operation of an individual PHY port and physical connection is independent of station type and topology. In the next section, we discuss the functional partitioning of the PHY port and the reasons behind the partitioning chosen. Subsequent sections describe the distributed clock scheme, the design of the physical link, and the impact of physical link errors on the LAN.

Functional Partitioning

In this section, we describe the partitioning of the functions of the PHY port into the following components:

- PHY (physical protocol chip also referred to as the ELM chip)
- CDCT (clock and data conversion transmitter)
- CDCR (clock and data conversion receiver)
- FOT (fiber-optic transmitter)
- FOR (fiber-optic receiver)

Our choices for the appropriate partitioning and technology were founded on our decision to develop a highly integrated and low-cost chip set. After examining several alternatives, we chose a partitioning that enabled us to use mostly CMOS technology (complementary metal oxide semiconductor), a minimal amount of custom ECL (emitter coupled logic), and no ECL gate array technology. Although a 125-megahertz (MHz) serial channel requires ECL circuitry in the system, we wanted to minimize the amount of custom ECL technology. ECL consumes a substantial amount of current

and is relatively expensive as compared to CMOS technology. We also considered ECL gate array technology, but decided against it because it was not a mature technology, lacked requisite analog functions for clock extraction, and was available from only a few vendors.

We determined that the CDCR and CDCT were the only functions that had to be implemented in ECL technology. This determination was based on the need for the high transmission rates and for quick conversion to and from serial and parallel data streams. The CDCR receives a 125-megabaud ECL serial data stream from the FOR. Using a phase lock loop, CDCR extracts a receive clock to recover the data bits and then converts the serial data to a 5-bit parallel bus. The CDCT receives a 25-MHz, 5-bit-wide parallel bus; then, by using a phase lock loop, it generates an internal 125-MHz transmit clock in phase with the local 25-MHz clock. CDCT then converts the 5-bit-wide parallel bus to a 125-megabaud ECL serial bit stream that is transmitted by the FOT.

We selected the 5-bit width for the parallel bus to obtain a 25-MHz bus rate. This rate is a convenient divisor of the 125-MHz serial rate and is within the operating range of the CMOS gate array technology used in the connected chips. The 5-bit bus also offered the advantage of enabling us to maintain a low pin count on the devices to which the bus is interfaced, thus further containing costs.

Another complication relative to the clock component was how to distribute a 125-MHz clock signal. As noted earlier, some FDDI products have many PHY ports, and those PHY ports must have a common clock line for the transmission of the serial data stream. We decided to add another phase lock loop in the transmit component that would lock onto the 25-MHz local clock, generate the 125-MHz serial clock, and convert the 5-bit parallel bus to the serial stream. With this method, the highest clock rate distributed on our boards was a 25-MHz clock.

As a consequence of selecting the transmit phase lock loop, we chose to specify and build separate transmit (CDCT) and receive (CDCR) devices in custom ECL technology. We were very concerned that the combination of two asynchronous phase lock loops on a single chip would induce cross talk. Cross talk could cause false locking of the phase lock loops to one another, resulting in lost data. Therefore making a single chip was considered too risky for the initial implementation. Our solution was to specify the transmit and receive devices, thus eliminating the possibility of cross talk.

The balance of the logic for the physical layer protocols could now be designed in CMOS gate array technology. The use of CMOS gate arrays was important in meeting schedule since it allowed us to quickly implement changes. Changes were inevitable and therefore had to be accommodated because the ANSI standard was not finished and stable during our design cycle.

All of the physical layer functions such as the encoder, decoder, elasticity buffer, framer, and smoother were implemented in CMOS gate arrays. We had simulated these functions in software; however, we were now able to build them using CMOS gate arrays and actually analyze their behavior in real networks. With the hardware, we quickly verified the protocols defined in the FDDI standards. Proper PHY operation is best confirmed by testing actual implementations.

The fiber-optic transmitter (FOT) converts a 125-megabaud electrical signal to light pulses to be transmitted to a receiving station. The fiber-optic receiver (FOR) receives the pulses from a transmitting station and converts them to an electrical data stream. We decided not to develop the FOT and the FOR components ourselves. Instead we chose to influence the specification of the system's functional requirements in the ANSI FDDI Committee and then depend on external vendors to develop the components.

It was important to encourage the optical vendors to standardize their components so costs would decrease, and so that more than one source of optical components would be available to us. Accordingly we did not combine the optical transmitter and receiver with any other physical layer functions. The optical link design is the subject of a later section in this paper.

Operation of the Distributed Clock Scheme

In the FDDI distributed clocking scheme, each station uses an independent, local clock reference when transmitting or repeating data packets. The station must synchronize the receive data with its own reference clock prior to further processing. Although this distributed clock reference scheme simplifies many problems, it also can give rise to data integrity problems and packet loss rate issues that must be solved for the scheme to work effectively.

Data must be synchronized to the local clock reference in a way that prevents detected and undetected errors caused by metastability problems. Further, interpacket gap shrinkage that can

result in an unacceptable packet loss rate must be controlled. In the sections Elasticity Buffer and Smoother below, we describe how these problems are addressed in the physical layer protocol.

Elasticity Buffer

Each PHY port of a station must accept data packets from another station with a slightly different clock frequency and bit transmission rate. It is the function of the elasticity buffer within the PHY port to synchronize the incoming data to the local clock reference. The buffer is also designed to control synchronizer metastability, a source of undetected data corruption. As a result of the elasticity buffer operation, the size of the gap between two data packets varies as the packets are repeated around the ring.

The elasticity buffer is a collection of storage registers that are written to and read from at different rates. (See Figure 3.) The buffer forms a circular queue due to the movement of two independent pointers: the input pointer selects the register to be written to and moves at the recovered clock rate; the output pointer selects the register to read from and moves at the local clock rate. The location of the pointers is based on the gray code counters. The input pointer is controlled by the input state machine and the output pointer by the output state machine. These state machines position the pointers at a controlled distance from one another. Therefore pointer control prevents data from being written while it is being read from the same register, even though the pointers are moving at different rates.

During normal operation, the input and output pointers approach each other and must be periodically repositioned. The repositioning occurs during the idle time between data packets, known as the interpacket gap. When a minimum interpacket gap time is detected by the input state machine, a reset control signal is sent to the output state machine. The reset signal is synchronized by the output state machine to avoid metastability. When an input signal to a register is changing at the time the register is clocked, its output may become indeterminate and assume multiple values over a time called a period of metastability.⁴ The reset signal could be changing when it is sampled by the output state machine, so this signal is synchronized by waiting an interval after each sample for the sample value to settle before the sampled value is utilized, or considered valid. The reset signal is delayed by this process. The circuitry guarantees that the

present address of the input pointer has been in the holding register on the reset condition for a sufficient duration and thus its stability is ensured. The output state machine then loads the address that the input state machine stored in the holding register. The output pointer moves to that location plus an offset in order to keep a minimum distance from the input pointer.

This approach to repositioning pointers ensures that all data or signals are stable in their respective registers before being sampled by the local clock. As a consequence, we were able to specify in the design that only one control line be synchronized in the elasticity buffer. The signal that crosses the clock boundaries is the reset signal, which is generated by the remote clock and sampled by the local clock. The reset signal triggers all events required for the elasticity buffer to correctly receive data. Since there is only one control line within the elasticity buffer that needs synchronization, the implementation is very robust.

We also had to anticipate occurrences outside normal operation. Therefore we designed circuitry that detects when the pointers point to the same

register for more than a minimum amount of time.⁵ This circuitry prevents the buffer from reading the register while its contents are changing; if the buffer were read, data corruption and consequently undetected errors could result during abnormal operation. The input and output pointers are 3-bit, asynchronous gray code counters. The pointers are compared to one another to determine whether they coincide, indicating that data in the buffer has overflowed or underrun.

The input and output pointer counters are compared using the local clock; that is, the input pointer counter bits can change with respect to the local clock, as shown in Figure 4. Gray code counters limit to one the number of bits that can change in the pointer counters at any sample interval. The comparator circuit is sampled twice using the local clock and the local clock shifted by 90 degrees. If one sample of the D flipflop notes a change, the changing bit settles down before the other sample happens; thus metastability problems are controlled. A logical AND of the sampled outputs signifies that the two pointers have had the same address for at least one quarter of the local clock interval. When

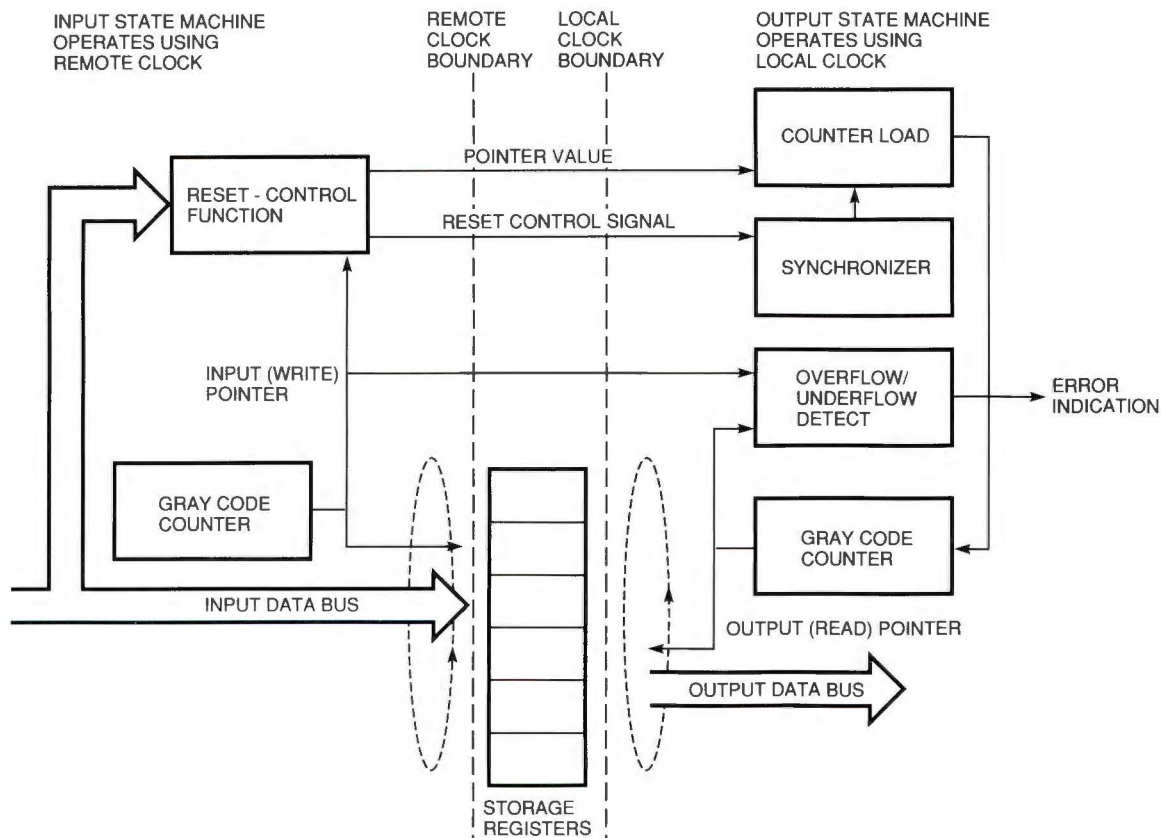


Figure 3 Pointer Control in the Elasticity Buffer

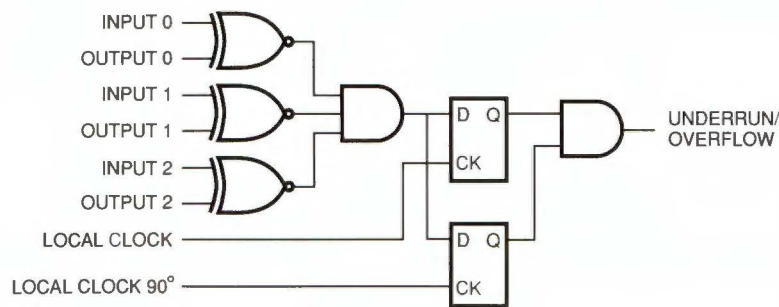


Figure 4 Overflow/Underflow Detection in the Elasticity Buffer

the addresses are the same for at least one quarter of the local clock time, the error flag is raised. When the error flag is raised, an overflow or underrun is imminent. The cause is somewhere in the network. For instance, the clocks are out of specification, or a downstream station is sending larger data packets than are permitted. In any case, the condition is detected and prevented from increasing the risk of undetected data corruption.

Smoother

Another function in the physical layer, called the smoother, prevents the loss of data packets that can result from shrinkage of the interpacket gap. Data packets can be lost, or discarded, at two places. The decision to discard can be made at the physical layer as the result of an elasticity buffer overflow or underflow. Also, packets can be discarded at the media access control (MAC) layer. The MAC layer is not required to copy a packet that has less than 6 idle bytes of interpacket gap preceding the packet.

The decision to implement the smoother came after simulations of the elasticity buffer revealed the then current draft ANSI PHY protocol would result in an unacceptable packet loss rate. In a series of nodes with a random distribution of clocks, some stations add to and others delete from the interpacket gap. If nodes add or delete without regard to the size of the interpacket gap, we found that the interpacket gap could be deleted entirely or reduced to a minimum size. At this size, the MAC is not required to copy a frame, and data packets are lost. Our simulation showed that an unacceptable 10 percent packet loss would occur due to 6 or less bytes of interpacket gap under these conditions:

- One hundred one elasticity buffers
- Maximum size data packet length
- Pseudorandom clock distribution

The solution is to monitor the interpacket gap.⁶ If it falls below the 7-byte minimum size, the smoother adds to the interpacket gap. The addition of interpacket gap to the output stream causes the elasticity buffer to use a buffer to delay the output data and then send an interpacket gap byte. The amount of buffering within the elasticity buffer is finite so that the delay within a station is not long. The smoother also reclaims storage elements by deleting bytes of interpacket gap one byte at a time from long preambles. This reclamation occurs any time 8 bytes or more of interpacket gap appear at a station.

The smoother is a distributed algorithm that cannot be adequately proven in simulation. We built a gate array that contained the elasticity buffer with the added smoother function to prove in hardware that our new algorithm would operate properly. We built a 200-node ring of elasticity buffers that could handle at least the test case (101 elasticity buffers) used in the simulation. Each elasticity buffer had a variable oscillator to allow control of the distribution of clock frequencies that we felt would induce interpacket gap shrinkage. We also included special test features in the chip to monitor the interpacket gap at every node in the tester.

During four weeks of ring operation, we observed the interpacket gap of 6.72 billion maximum size packets (4500 bytes). The minimum interpacket gap observed was 7 bytes, which resulted in no packet loss. The experiment indicated a packet loss rate of less than $2E-10$. Since no packet loss occurred, the actual loss rate is unknown; but this result gives us confidence that the loss rate predictions made by analysis are correct.

The 200-node hardware test bed demonstrated that our algorithm worked effectively. The smoother protocol was adopted as a mandatory part of the final ANSI FDDI PHY standard.² The standard allows

a variety of different designs, some having as little as one byte of smoothing, depending on the number of preamble bytes required by the MAC implementation in the same station. All allowed designs have worst-case loss rates below $1\text{E}-14$ (by analysis) in a homogeneous ring. The worst-case packet loss rate in a heterogeneous ring, one with multiple types of smoother designs, has a packet loss rate below $1\text{E}-10$ (by analysis). Given the addition of the smoother to the PHY protocol, the FDDI distributed clocking scheme does not significantly contribute to packet loss.

FDDI Optical Link Design

As noted in the earlier section, Operation of the Physical Layer, the physical connection is the basic element in the FDDI LAN topology. The physical connections between FDDI stations are full duplex, fiber-optic links that deliver a serial code bit stream from one station to another with a bit error rate (BER) less than $2.5\text{E}-10$. Each station has a separate transmit and receive link, and both links are cabled together to the same destination. The optical link requirements are defined and measured at the MIC. Any set of conforming FDDI stations connected together with a compliant cable plant in a legal topology are guaranteed to provide the required transmission service. Conformance to the optical requirements can be measured independently of both the interconnecting media and the attached station. Measurements can be taken from either end of a physical connection.

The technology choices we confronted and the design methods we used in the development of the optical link are summarized in the following sections. These methods can be applied to any transmission system design problem with similar requirements. The Physical Layer Medium Dependent (PMD) Working Group of the FDDI committee adopted these methods, and Digital played a leading role in the design of the PMD Standard. The Working Group developed the design in a manner that combined theoretical analysis with empirical modifications in an iterative process to arrive at the specifications for the system. The full detail of the models has been documented previously in the literature.^{7,8}

Technology Choices

The FDDI distance and bit rate requirements clearly mandate the use of a fiber-optic transmission system. However, the choices are not equally obvious between laser- or LED-based transmitters, between 850-nanometer (nm) and 1300-nm operation, and

between single-mode and multimode fiber operation. FDDI development initially focused on the transmission distance requirements of LANs which serve as local office networks as well as network backbones. Accordingly the technology chosen for the optical link should be cost-effective and capable of spanning approximately 2-km distances. For these applications, the superior bandwidth and loss characteristics of 1300-nm LED systems prevail over 850-nm LED technology; 125-megabaud transmission over 2 km is not possible with 850-nm LEDs. LEDs and multimode fiber prevailed over lasers operating with single-mode fiber because at the design time the former was more reliable and had a better chance of achieving the cost goals required by short-distance office interconnection spans. The selection of appropriate technology was especially difficult because the technology was rapidly evolving. The Working Group made the basic technology choices in the 1984-1986 time frame; the chosen technology represented the best compromise between available technology and reasonable anticipated improvements. The FDDI committee later addressed the long-distance requirements (greater than 2 km) of a campus LAN with a single-mode fiber and laser transmitter PMD (SMF-PMD). That development effort is not addressed in this paper.

Optical Link Overview

The optical link is composed of three basic elements: a transmitter, a cable plant, and a receiver. The transmitter is provided with a serial 125-megabaud code bit stream and creates an amplitude modulated 1300-nm optical version of the bit stream. The code bit stream has previously been encoded with a 4-bit into 5-bit (4B/5B) non return to zero invert (NRZI) coding scheme that ensures that the serial sequence has sufficient transitions to allow recovery of the transmit station's timing clock at the distal end of the link. The cable plant uses a glass, graded-index, multimode optical wave guide to ferry the signal to the receiver; the cable has an arbitrary number of junctions (e.g., connectors). The cable plant is described by its optical loss and bandwidth. The receiver in turn converts the optical signal back into a logic-level code bit stream. When a station is not sending data, the transmitter is provided with special code bit sequences which ensure that there is always an optical signal on the medium between packet transmissions. Thus, whenever the link is a part of a ring, the optical system stays in its equilibrium operating conditions, and the clock recovery circuit is always synchronized with the incoming stream.

Design Methods

The design of any digital transmission system must provide sufficient end-to-end bandwidth and signal power. Further, the design must demonstrate bounded jitter characteristics in order to provide the required data transfer at the desired BER. The bandwidth allocation, jitter budget, and loss budget for the FDDI optical system are described next.

Bandwidth Allocation and Models Nyquist communications theory requires a system bandwidth of at least one half the baud rate to prevent error rate degradation due to intersymbol interference. Practical systems require a somewhat greater bandwidth. We determined the LED-fiber bandwidth for FDDI by measuring the sensitivity of commercial 125-megabaud optical receivers as a function of increasing input rise time (decreasing bandwidth) and by observing when the channel bandwidth started to cause a penalty in the measured receiver BER performance. The 0.5 decibel (dB) optical power penalty point was found at 95 MHz. That point is the bandwidth requirement for the LED and fiber combination in a worst-case maximum length link; lower bandwidth causes increasingly higher penalties in BER performance and must be prevented.

The bandwidth of an LED and multimode fiber optical system is modeled with three components which add in a root mean square (RMS) fashion as shown in equation 1 in Figure 5. The design problem confronted is as follows: how are the three different bandwidth components rationally allocated to meet the 95-MHz LED-fiber requirement, and what is the maximum distance that can be achieved and still meet this requirement? Although the electrical and modal bandwidth limitations are well known (equations 2 and 3 in Figure 5), the chromatic bandwidth limitation caused by the LED and fiber combination was not well understood.

Chromatic bandwidth limitation is caused by the interaction of the LED spectral width with the wavelength dispersion of the glass fiber. Thirteen hundred-nm LEDs are not monochromatic; their emission spectrum is typically 170-nm wide at the half optical power point. The propagation velocity of light in glass is a function of the wavelength of the light; light of different wavelengths experiences differential delay or dispersion. Accordingly a signal of appreciable optical spectral width experiences dispersion that causes an increase in the signal transition times and limits the bandwidth. The amount of dispersion experienced by a pulse is a function of the length of the fiber, of the optical spectral width, and of the separation of the pulse

central wavelength from the zero dispersion wavelength of the fiber. The wide spectral width of 1300-nm LEDs is sufficient to cause systems based on their use to be distance limited by chromatic dispersion, even though the system is operating at 1300 nm, which is the nominal zero dispersion window of fiber. A model was developed and verified for the chromatic bandwidth; the equation for the model is 4 in Figure 5.

Equations 1 through 4 are the complete model for the bandwidth of the FDDI optical system. The inputs to the model are the transmitter spectral center wavelength, spectral width, transmitter rise and fall times, the fiber length, the fiber modal bandwidth, the fiber's zero dispersion wavelength (λ_0), and the zero dispersion slope (S_0). These parameters completely define the constituents of

$$BW_{sys}^{-2} = BW_{ele}^{-2} + BW_{mod}^{-2} + BW_{chr}^{-2} \quad (1)$$

Where:

BW_{sys} = total optical system bandwidth (MHz)

BW_{ele} = electrical bandwidth of LED (MHz)

BW_{mod} = fiber modal bandwidth (MHz)

BW_{chr} = chromatic bandwidth of LED-fiber (MHz)

$$BW_{ele} = \frac{470}{T_{rf}} \quad \text{MHz} \quad (2)$$

$$BW_{mod} = \frac{MBW}{l} \quad \text{MHz} \quad (3)$$

$$BW_{chr} = \frac{375}{l \cdot \Delta\lambda \sqrt{D^2 + \frac{(S_0 \cdot \Delta\lambda)^2}{8}}} \quad \text{MHz} \quad (4)$$

$$D = S_0 \cdot ((\lambda_c - 7) - \frac{\lambda_0^4}{(\lambda_c - 7)^3})$$

Where:

T_{rf} = LED rise or fall time (ns)

MBW = modal bandwidth distance product of the fiber (MHz · km)

l = the fiber length (km)

$\Delta\lambda$ = 0.85 · LED Full Width Half Maximum (nm)

S_0 = the fiber dispersion slope (ns/nm² · km)

λ_c = the LED central wavelength (nm)

λ_0 = the fiber zero dispersion wavelength (nm)

Figure 5 FDDI Optical Link Bandwidth Model

the bandwidth of a multimode fiber-optic transmission system. In an iterative sequence of calculations with the model, we evaluated a trade-off of fiber length, fiber modal bandwidth, LED chromatic attributes, and LED rise and fall times to arrive at a 2-km maximum fiber length with transmitter chromatic and temporal requirements that could reasonably be met by vendors. The transmitter requirements were described by a series of curves that balanced transmitter rise and fall times and chromatic attributes. These requirements guarantee the 95-MHz LED-fiber bandwidth requirement for a 2-km fiber. Thus transmitters are allowed slow rise times if they have narrow spectral widths or central wavelengths that match the minimum dispersion wavelength of the fiber. Transmitters with wider spectral widths and central wavelengths displaced from the zero dispersion wavelength have fast rise time requirements. The curves in ANSI FDDI PMD Figure 9 show the final allowed transmitter spectral and temporal trade-offs.³ They were generated with a slight modification to the basic model described above that used explicit fast Fourier transform (FFT) descriptions of the LED electrical bandwidth component. The transmitter requirements depend on the fiber meeting modal bandwidth and chromatic dispersion specifications. We empirically established the 500 MHz·km minimum modal bandwidth distance product requirement and the allowed range of dispersion parameters shown in the ANSI PMD Figure 14.³

Jitter Budget In most high-speed serial digital communications systems, the clock used to recover the received data must be extracted from the bit stream. The recovered clock is used to sample the data, and the sampling transition is nominally in the middle of the bit interval. If the sampling clock location overlaps with the signal transition between bits, errors occur. Jitter is time dither of the bit stream signal transitions; the measured value is a function of the probability of its occurrence. Because jitter is the predominant source of communications system error, it is measured at a probability equal to the BER requirement.

A jitter budget tracks the accumulation of jitter in the bit stream edge position and allocates it to different components. The budget ensures there is a jitter-free opening, or window, for the placement of the sampling clock. Jitter consists of three basic types:

- Duty cycle distortion—DCD
- Data dependent jitter—DDJ
- Random jitter—RJ

DCD is static and is caused by switching threshold variation and mismatched rise and fall times in driver circuits. DDJ is caused by bandwidth limitations in transmission components and is also a function of the transmitted code bit stream. We developed a worst-case test pattern that evinces high-frequency DDJ components caused by local run length variations in the transmitted bit stream and low-frequency DDJ components caused by variations in the average power of the unbalanced 4B/5B code bit stream. RJ is caused primarily by thermal noise corrupting the signal in receivers and is apparent at low optical powers. RJ adds in a root-mean-square fashion with other RJ components; DDJ and DCD add linearly to RJ.

The FDDI jitter budget tracks these three components of jitter through the optical link. The budget ensures a sufficient allocation for the clock recovery implementation to place the clock correctly in the jitter-free window to retiming the data. The specific values of jitter allotted to each link element were determined largely by empirical methods. The sum of all jitter allocations must not exceed the code bit width [8 nanoseconds (ns)]. Table 1 summarizes the jitter budget, showing the totals for each jitter component as it adds through the link.

Only the jitter components visible at the PMD MIC (PMD out and PMD in) are enforceable parts of the standard. Note the sum of the jitter components at PHY in (the exit of the receiver function) is 5.87 ns, leaving a 2.13-ns jitter-free window remaining in the 8-ns bit cell. This window is allocated to the static alignment error and RJ of the clock recovery implementation. Digital developed specialized test equipment to generate and receive the DDJ test pattern and to signal received bit errors; the error rate at the worst-case optical conditions (minimum power, maximum jitter) was measured as a function of clock sampling position to measure the jitter-free window at the receiver exit. The 2.13-ns jitter-free window is the measured receiver component requirement.

Table 1 FDDI Jitter Budget Example (Nanoseconds Peak to Peak)

Measurement Point	DCD	DDJ	RJ
PHY out	0.4	0.0	0.32
PMD out	1.0	0.6	0.76
PMD in	1.0	1.2	0.76
PHY in	1.4	2.2	2.27

Optical Loss Budget The optical loss budget for FDDI is the difference between the minimum optical power launched into the fiber and the minimum optical power required at the receiver. Decreased optical power in a receiver causes a reduction in the signal-to-noise ratio and is evinced on the serial data stream as an increase in its RJ. The optical power requirements are defined in terms of the performance measured with 62.5-micron-core multimode fiber. The fiber core size is specified because the launch power of a particular transmitter is a function of the fiber type used. This fiber type is the prevalent standard for fiber-optic LANs and with it an 11-dB loss budget is provided for the optical link.

The 11-dB loss budget is apportioned by a user between bulk fiber losses (1.5 dB/km typical, 2.5 dB/km worst case), connector losses (0.6 dB typical, 1.0 dB worst case) and splice losses (0.2 dB typical, 0.5 dB worst case). With this loss budget, users can construct cable plants of up to 2 km in length with any number of connectors and splices, provided the total loss is less than 11 dB. There is no minimum loss required because the maximum launch power is equal to the maximum input power; stations may be operated back to back without saturating the receiver function.

In summary, the design methods we used guaranteed the bit error rate of the serial data stream transmission between stations. The optical bandwidth was allocated and guaranteed by design to prevent BER degradation due to intersymbol interference; the jitter accumulation from different link elements was budgeted to prevent BER degradation due to received data sampling errors, and an optical power budget was defined to control BER degradation due to inadequate receiver signal-to-noise ratio.

Physical Link Error Process

An important part of the physical layer development was the analysis of the media bit error processes. In the previous section, we presented the design of the optical link to control the bit error rate. This section considers the effect of the error process and the isolation of certain types of faults that cause errors.

To evaluate the error process, we had to know the source of the errors and study their effect on the protocols for the FDDI. The error process must be considered in light of two metrics:

- Correctness of the protocol. Error events may lead to undetected corruption of user data. Detected errors reduce performance, but an undetected error may have nearly unbounded bad effects for the user. Therefore the undetected error rate must be very low.

- Isolation of error source to a component of the network when the error rate is too high. Error rates may exceed acceptable levels as the result of a misconfigured network or a fault. Isolation of the problem is the first step in a repair process.

A discussion of protocol correctness and fault isolation must consider more sources of errors than the normal bit error process discussed in the previous section. To provide correctness and fault isolation, the design must account for misconfigured links and common faults. A misconfigured LAN may provide poor performance, but it is always unacceptable for a data packet to be delivered with undetected errors.

Good examples of misconfigured links include those with cables that are too long and that use too many connectors in the cable plant. Common faults in the system include transmitters that are too dim, dirty or partially plugged connectors, and cables that are kinked (for example, by a misplaced chair leg). One can write an endless list of possible faults and can posit a fault with an arbitrarily complex symptom. The faults listed above are important because they are likely to occur during normal use of the components. Many of these faults can be traced to a careless or uninformed user. A design must ensure that these external causes of abnormal error rate do no lasting damage and that they can be detected and isolated.

The error process resulting from important faults is similar to the error process of a correctly operating optical link. As discussed earlier for the design of an optical link, bit errors are caused by transition jitter resulting from bandwidth and power budget limits. The important faults and misconfigurations reduce the channel bandwidth or increase the optical loss beyond the design limits. The error rate may exceed the design limit but the physics of the error process remains the same.

We analyzed the impact of this error process given the FDDI encoding/decoding and error-detecting protocols. An example of an error event is shown in Figure 6 to illustrate the effect of media noise on the FDDI encoding schemes. The code bits on the media are encoded as NRZI, where a signal transition represents a code bit 1 and a lack of transition (for a bit time) represents a code bit 0. With this encoding, a single noise event results in two code bit errors where the resulting pair of bits are the complement of the original bits. In Figure 6, the pair of code bits 0,1 are changed to 1,0. There are four possible pairs of code bits—00, 10, 01, 11—that change to 11, 01, 10, 00, respectively, by an error

event. The FDDI PHY uses a block code in which 5 code bits represent a symbol, and a symbol contains 4 data bits. In the example, the single error event changes 2 code bits, which results in a decoded symbol with 4 incorrect data bits. The number of data bits affected by an error event is multiplied by the decoding process.

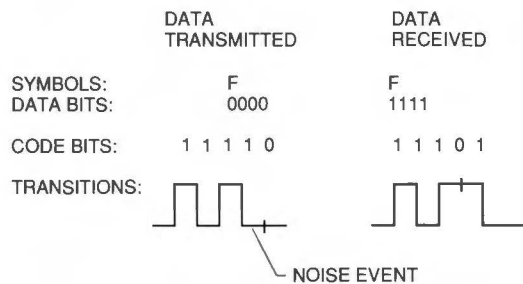


Figure 6 Example of a Single Noise Event

Error detection is provided by redundancy in the data packet. Errors are detected by the MAC protocol based on a frame check sequence (FCS). The probability of an undetected error is related to the number of error events in the packet and to the specific symbols created. Our analysis, based on a draft of the FDDI MAC protocol, indicated that undetected data corruption could occur with high probability.⁹ In the important case, a new frame was created when a noise event changed a data symbol into an ending delimiter and created a smaller frame. This truncation process resulted in an undetected packet error rate of $3\text{E}-14$ for large rings (500 stations).¹⁰ Our design requirements include the much more strict limit of $1\text{E}-21$ on this rate. For this reason, an enhancement to strengthen the ending delimiter was proposed and accepted by ANSI X3T9.5 for the MAC protocol.¹¹ In accord with this enhancement, a frame is valid only if its ending delimiter is followed by a symbol that cannot be created by the noise event that created the ending delimiter. Thereby, undetected corruption was greatly reduced in the final, standard MAC protocol.¹² This enhancement results in a undetected error rate of $5\text{E}-24$ for the protocols, allowing significant margin for actual implementations.¹⁰

To isolate a faulty physical link, we need to know which of many links exceeds the design-specified error rate. Each error event must be detected and counted at one point in the topology. Using a traditional method, we would isolate faults based on the information provided by the MAC FCS error counters.

Although this method works reasonably well for a bus topology, it is more difficult to use with FDDI topologies. The quantity of physical links may greatly outnumber the MACs in the topology. The errors from more than a single physical link may be counted by one MAC, thus masking which links exceed the error rate. For example, in a wrapped dual ring of single-MAC, dual attachment stations, data errors occurring in only half the physical links in the network would be counted by a single event counter. A similar situation occurs in an FDDI tree topology. The MAC error counters are not associated with a particular physical link.

Fault isolation must be based on facilities present for each physical link. For this purpose, we developed a protocol called link error monitor (LEM). LEM takes advantage of the requirement in the PHY standard that a set of code bit groups representing violation symbols and certain sequences of control symbols not be transmitted (repeated) onto a physical link. Our study of the error process indicated that roughly 30 percent of the error events could be detected by the physical layer decoder.¹⁰ This accuracy is acceptable as BER variations of many orders of magnitude are often the most important. LEM counts the decode violations that are received only at one point in the LAN immediately after the error event occurs. Errors not counted by LEM are those in which the created symbol may be repeated by a PHY port, such as when a data symbol is changed to another data symbol. An instance of LEM protocol may observe each PHY port and detect events associated with a particular physical link.

The accuracy of a LEM BER estimate is comparable to other methods and has the advantage of providing better fault isolation. The accuracy of a LEM estimate is affected by the statistics given above for the error process and the length of packet transmitted on the ring. Generally we only assign significance to the order of magnitude of the estimate, i.e., the exponent of the BER written in scientific notation. This type of accuracy problem is shared by BER estimates based on MAC FCS error counters as well. For instance, the FCS-based estimate of BER also depends on packet length and additionally on ring utilization. The FCS error counters count errors in valid packets only, so estimates of error rate are strongly affected by ring utilization. The LEM estimate includes error events in tokens, stripped frames, and those that occur during the idle period between packets.

The LEM protocol counts errors and provides a BER estimate for each link in the FDDI LAN. Network

management applications may collect this data and identify marginal links within the LAN. In addition, LEM provides a mechanism to automatically eliminate faults from the network. This fault-recovery procedure preserves the integrity of the ring when physical links would otherwise prevent ring operation. The LEM protocol was proposed and included in the draft FDDI Station Management (SMT) proposed standard.¹³

The analysis of the physical layer error process resulted in two important changes that reduce the impact of errors. A change proposed and adopted in the FDDI MAC protocol greatly reduced the rate of undetected corruption. The isolation of components contributing to a high error rate is facilitated by LEM, now a part of the draft FDDI Station Management standard. These developments have improved the correctness and maintainability of the FDDI LAN.

Summary

Our development work on the FDDI physical layer provided physical layer components, specifications and new protocols. This paper has described the operation of the FDDI physical layer and the functional partitioning of the chip set. The functional partitioning resulted in greater integration and lower cost for the chip set. Much of the work on the physical layer centered on the need to control the error characteristics of both the constituent links and interplay of many asynchronous links as a system. Three important design problems were solved during the development effort. First, the elasticity buffer and smoother protocols, which were developed for the distributed clocking scheme, resolve data integrity and data loss problems. Second, the design of the fiber-optic link for FDDI required methods to allocate system bandwidth and power margins. The bandwidth, jitter, and loss budgets provided a means to allocate channel margin between individual components and can be applied to the design of many transmission systems. Finally, the analysis of the physical link error process resulted in increased correctness through a reduction of the undetected error rate and enhanced fault isolation provided by the link error monitor, LEM.

Acknowledgments

The authors have reported on the results and work of many individuals. Special acknowledgment is extended to Raj Jain, Don Knudson, Charlie Kaufman, and Herman Levenson for their original contributions to the development and for their help in writing this paper.

References

1. W. Hawe, R. Graham, and P. Hayden, "Fiber Distributed Data Interface Overview," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 10-18.
2. *FDDI PHY, ANSI X3.148-1988, Token Ring Physical Layer Protocol* (New York: American National Standards Institute, 1988).
3. *FDDI PMD, ANSI X3.166-1990, Physical Layer Medium Dependent* (New York: American National Standards Institute, 1990).
4. T. Chaney and C. Molnar, "Anomalous Behaviour of Synchronizer and Arbiter Circuits," *IEEE Transactions on Computers* (April 1973): 421-422.
5. B. Thompson and J. Iannarone, Method and Apparatus for Detecting Impending Overflow and/or Underrun of Elasticity Buffer, U.S. patent no. 4,945,548.
6. C. Kaufman, M. Kempf, and J. Hutchison, Method and Apparatus for Nodes in Networks to Avoid Shrinkage of an Interframe Gap, U.S. patent no. 4,878,219.
7. J. Hutchison and D. Knudson, "Developing Standards for a Fiber Optic LAN-FDDI," *SPIE Proceedings*, vol. 715 (1986).
8. D. Hanson and J. Hutchison, "LED Source and Fiber Specification Issues for the FDDI Network," *IEEE Computer Society Proceedings COMPCON* (1987).
9. Draft of proposed standard for FDDI MAC, Accredited Standards Committee (ASC) X3T9.5/83-16, Token Ring Media Access Control, Rev. 10 (1986).
10. R. Jain, "Error Characteristics of Fiber Distributed Data Interface," *IEEE Transactions on Communications*, vol. 38, no. 8 (1990): 1244-1252.
11. R. Jain, "FDDI Error Analysis," FDDI X3T9.5 Working Group on SMT, Committee Document SMT-80 (1987).
12. *FDDI MAC, ANSI X3.136-1987, Token Ring Media Access Control* (New York: American National Standards Institute, 1987).
13. H. Levenson, "Link Error Monitor (LEM)," FDDI X3T9.5 Working Group on SMT, Committee Document X3T9.5/88-251 (October 1989). See also "LEM Error Monitor," Document X3T9.5/88-198 (August 1989).

FDDI Data Link Development

The fiber distributed data interface (FDDI) data link is based on the ANSI X3T9.5 FDDI standards with Digital's enhancements to provide greater performance, reliability, and robustness. The FDDI project team encountered significant challenges, including the evolving ANSI X3T9.5 FDDI standards and the development of the technology to implement the data link, coupled with time-to-market pressure. Appropriate considerations and design trade-offs were made to design complexity, performance, risk, cost, and schedule, when deciding functional partitioning and semiconductor technology. Extensive simulations and a novel test approach were used to verify the algorithms, the functional models comprising the chips, and the physical chips themselves.

The proliferation and importance of distributed system applications place special requirements on networks in terms of topological flexibility, performance, reliability, scalability, robustness, and efficiency. The advent of fiber optics, large-scale integrated circuits, and related technologies makes it possible to provide a relatively low-cost, high-speed local area network (LAN) with large physical extent and connectivity. One LAN standard is the fiber distributed data interface (FDDI), a 100-megabit-per-second token ring that uses an optical fiber medium.

The scope of FDDI spans the data link layer and the physical layer. The FDDI data link provides its users with communication services on a multi-access LAN for transmitting and receiving frames with best-effort delivery service (also called the datagram service). The development of the FDDI data link encountered several significant challenges, including the instability of the standard, unproven technology and protocols, and an order of magnitude increase in speed from the International Standards Organization (ISO) 8802-3 carrier sense multiple access with collision detection (CSMA/CD) LAN. The design of the FDDI data link involved performance considerations such as throughput, latency, data integrity, and reliability.^{1,2}

In this paper we discuss the development of Digital's FDDI data link and present some of the key algorithms developed by Digital. We then describe the design and development of the FDDI data link technology and its implementation in the FDDI, focusing on the FDDI data link's very large-scale integration (VLSI) chip set. Finally, we describe the

development methodology used in simulation, verification, and testing.

FDDI Data Link Overview

The FDDI data link provides an upward multiplexing, datagram service to support multiple data link users concurrently within the same computer system. The FDDI data link incorporates the ISO 8802-2 logical link control (LLC) and FDDI standards. Also, the FDDI data link provides a mapped Ethernet service, defined by the Internet RFC 1103 standard, to map an Ethernet frame onto an ISO 8802-2 LLC frame for transport over the FDDI LAN.³

The FDDI data link consists of an LLC sublayer, a media access control (MAC) sublayer, and their management. The LLC sublayer provides LLC services, the mapped Ethernet service, and multiplexing/demultiplexing services for multiple users. The key functions provided by the MAC sublayer include the FDDI token ring protocol, frame transmission and reception, initialization and error recovery for the token ring, address recognition and filtering on receive, and frame error detection. Figure 1 is an example of the FDDI architecture model, showing a dual attachment station (DAS) or dual attachment concentrator (DAC) with a single data link entity. A DAS or DAC may have zero, one, or two link entities; two or more physical (PHY) port entities; and control of their management. A link entity is an instance of data link that contains an LLC and a MAC entity. A data link user accesses the data link services through the port entity. As shown in Figure 1, multiple data link users may use the same link entity.

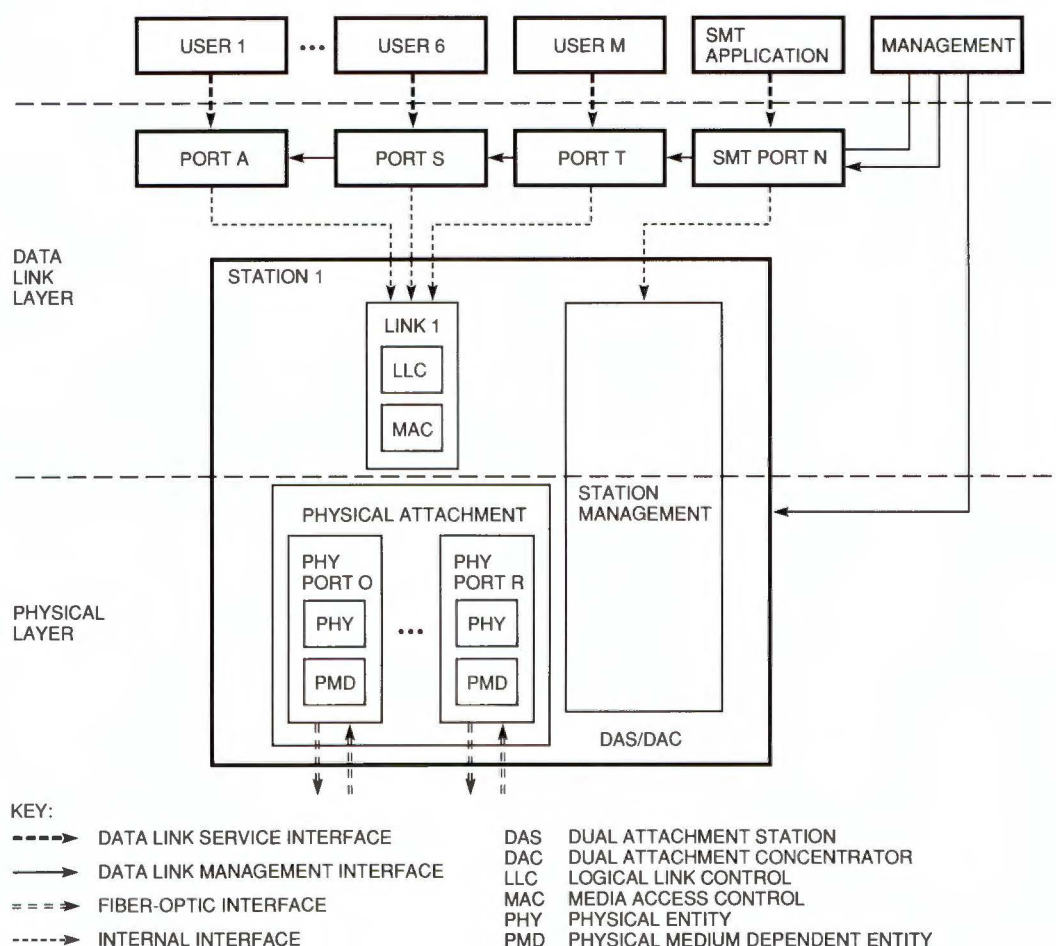


Figure 1 An Example of the FDDI Architecture Model

The FDDI data link uses the FDDI MAC protocols to provide fair access to a multiaccess channel, which is built of individual point-to-point physical links.⁴ A token ring consists of an ordered, cyclic set of MAC protocol entities called MACs. It operates by passing a token sequentially from MAC to MAC around the ring. Only the MAC with the token may transmit frames onto the ring, and only one token can be present on the ring at any instant. At the end of transmitting its frames, the MAC transmits the token onto the ring. Frames circumnavigate the entire ring and are subsequently removed (stripped) by the originating MAC after one and only one rotation. The FDDI MAC protocol is different from the ISO 8802-5 token ring protocol.⁵

The FDDI MAC protocol uses a timed token protocol, whereby MACs on the token ring cooperatively attempt to maintain a specified token rotation time by using the observed network load to regulate the

amount of time a MAC may transmit.⁶ The specified token rotation time is called the target token rotation time (TTRT). The TTRT is negotiated using a distributed algorithm called the claim token algorithm, which is invoked each time the ring is initialized. The FDDI MAC protocol also includes fault detection and recovery functions to aid in the restoration of ring operation in the presence of transient faults.

As shown in Figure 1, each instance of a station (e.g., DAS or DAC) contains a set of station management functions. Some of the key station management functions included are initialization, observation and control of link and PHY port entities, control of the insertion and removal of the station from the ring, topology control, fault detection and recovery, and a set of frame-based protocols. The set of frame-based protocols for station management includes duplicate address detection,

neighbor notification protocol for ring map generation, and loopback protocol. The scope of the station management frame-based protocols is limited to a single ring. These functions are implemented in the Common Node Software (CNS) and the FDDI chip set.⁷

FDDI Data Link Algorithms

Digital's realization of the FDDI data link includes major enhancements and value-added features. The development of the FDDI data link encountered several architecture and implementation issues. The key issues included design for high performance, error characteristics and data integrity, removal of frames by bridges, cleaning the ring of unwanted frames and long fragments, and stability and reliability of the ring.² Digital provided the impetus to resolve these issues as well as the solutions to be incorporated into the ANSI FDDI standards. Some of these solutions are described below.

Frame Content Independent Stripping (FCIS) Algorithm

The underlying logical topology of all token rings is a closed loop structure, which inherently has the property of continuously circulating frames transmitted on the ring. Because of this property, all token rings have algorithms for removing frames transmitted by MACs on the ring. The frame removal algorithm is called a frame stripping algorithm. When stripping a frame, the fragment size (remnant of the frame) must be less than 17 bytes. Frames that are not properly stripped can remain and traverse the ring repeatedly, wasting bandwidth and resources within systems on the ring and causing severe congestion in the systems due to delivery of duplicate frames.

The FDDI MAC protocol uses a frame stripping algorithm in which all MACs on the ring continually strip received frames whose source address matches their own MAC address. Limitations of this algorithm arise when implementing bridges or systems that need to transmit frames with source addresses which are different from each MAC's own address.⁸ For example, a bridge may forward frames with no modifications, and, therefore, the forwarded frames contain source addresses which are different from the bridge address. Also, a bridge may support tens of thousands of stations in the extended LAN.⁹ If a bridge were to use the source address match algorithm, the bridge would have to complete the address match operation within one microsecond from the beginning of the frame reception.

Therefore, the use of the source address match algorithm by a bridge imposes significant costs and implementation complexity. The design of the frame stripping algorithm is made more difficult by the fact that there can be up to 560 frames outstanding (i.e., transmitted but yet to be stripped) and the fact that there is less than one microsecond to decide whether to strip or to repeat the frame.

Digital developed an algorithm called the frame content independent stripping (FCIS) algorithm, which is implemented in the MAC chip.¹¹ The algorithm is based on stripping the same number of frames that the MAC transmitted on the ring independent of the content of the frame. After the MAC captures the token for frame transmission, a local count is incremented each time a frame is transmitted. After transmitting all its frames, the MAC transmits a void frame, which is a minimum size frame (i.e., 17 bytes), before transmitting the token.¹² On receive, if the count is greater than zero, the received frame is stripped; and for each error-free frame received and stripped, the count is decremented. After transmitting the token, the MAC does not strip frames when the count is equal to zero, except for frames with a source address matching the MAC's address. When receiving an error-free void frame with the source address matching the MAC's address, the count is unconditionally reset to zero. As a result, the algorithm uses a count that is kept locally to track the number of outstanding frames for stripping, and it uses a transmitted void frame as a backup mechanism to indicate the end of stripping.

The operation of the FCIS algorithm is seen in the space-time diagram of Figure 2, which shows three stations on the ring while station B is the only station participating in the FCIS algorithm. Time increases from top to bottom, and the position of downstream stations on the ring goes from left to right in this diagram. Station A first receives the token T and transmits frames A1, A2, and the token. Station B then transmits frames X1, Y1, and Z1, which have source addresses X, Y, and Z that are not the same as station B's MAC address. As shown in Figure 2, each time station B transmits a frame, it increments its count; and each time station B strips an error-free frame, it decrements the count. After transmitting frame Z1, station B transmits its void frame, VB, and then the token. When station B receives its void frame, the count is reset to zero, which causes station B to stop frame stripping. Subsequently, an entirely new epoch of transmission and frame stripping can begin with the next token capture.

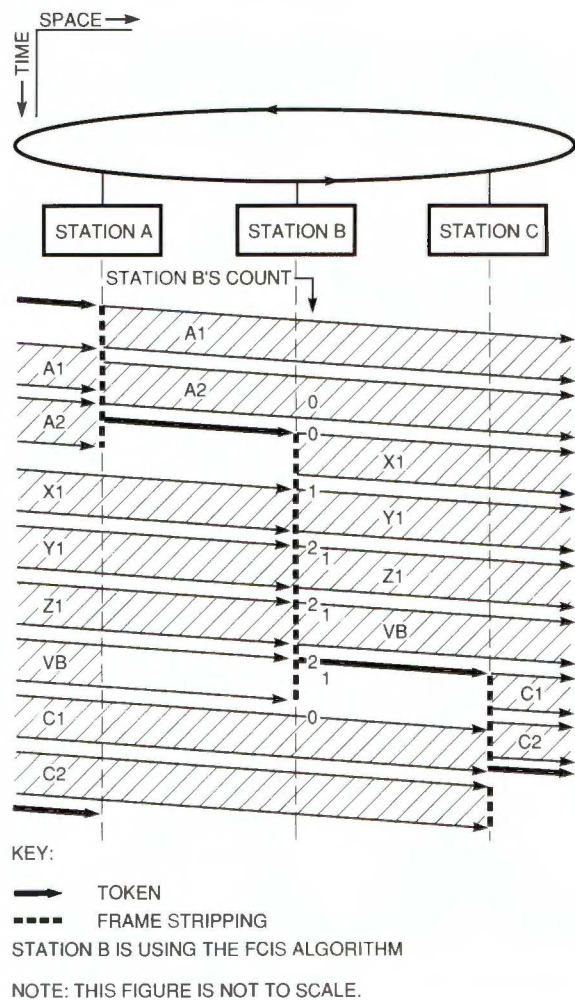


Figure 2 Frame Content Independent Stripping Algorithm Example

The combination of the count and the void frame provides robust frame stripping that is independent of the content of the transmitted frame. The FCIS algorithm also has the desired property of operating transparently with other MACs which are not implementing the algorithm. Implementing the FCIS algorithm in the MAC chip greatly reduces the cost and complexity of products, which otherwise may need additional hardware components to perform similar functions.

Ring Purging Algorithm

One of the well-known problems on a token ring is the circulation of frames or long fragments that are not stripped by the transmitter. The frame or long fragment not stripped by the transmitter is called a no-owner frame (NOF). On an idle ring, NOFs can

circulate around the ring, along with the token, continuously at the speed of the ring. NOFs may be received by one or more systems on the ring repeatedly at an extremely high rate, which can lead to severe congestion and waste of system resources. For example, a single NOF on an idle ring can create a frame arrival rate of 2,700 frames per second (for maximum size frame) to about 290,000 frames per second (for minimum size frame). In the best case, the NOFs are removed when they arrive at a MAC that is transmitting (i.e., holding the token).

Digital developed an algorithm called the ring purging algorithm to remove NOFs. The ring purging algorithm ensures that NOFs do not traverse the ring more than twice. The implementation of ring purging consists of two related, but different, algorithms. The first one is the purger election algorithm, which is a distributed election algorithm to select a designated MAC to be the purger for the ring. The second algorithm is the purging algorithm, which is executed by the designated MAC to clean the ring of NOFs. We describe the purging algorithm in this section.

The purging algorithm adopted for Digital's FDDI data link purges the ring transparently each time a token is received by the purger. When the purger receives a token, it begins a purge cycle by transmitting two special frames, called void frames. If the purger has frames to transmit, it completes the transmission of its frames before starting the purge cycle. Once the purge cycle has started, the purger unconditionally removes all frames or fragments received. The purge cycle is terminated when the purger receives one of its error-free void frames, a token, or a ring initialization frame. In order to increase the probability of correctly terminating the purge cycle, the purger transmits two void frames; but it terminates its purge cycle based on receiving only one error-free void.

Figure 3 shows the operations of the ring purger in removing an NOF, during an idle ring and during a busy ring. Each time station S3 (the purger) receives the token, it may transmit its frames, followed by two void frames and then the token. It purges the ring until it receives one of its error-free void frames. As shown in the example, the NOF was purged by station S3 on its second traversal around the ring. Also, the example shows that the purging of the ring is transparent (i.e., there is no disruption to the ring).

The impact of ring purging on ring performance is negligible, because the ring purger only initiates the purge cycle when it has the right to use a token.

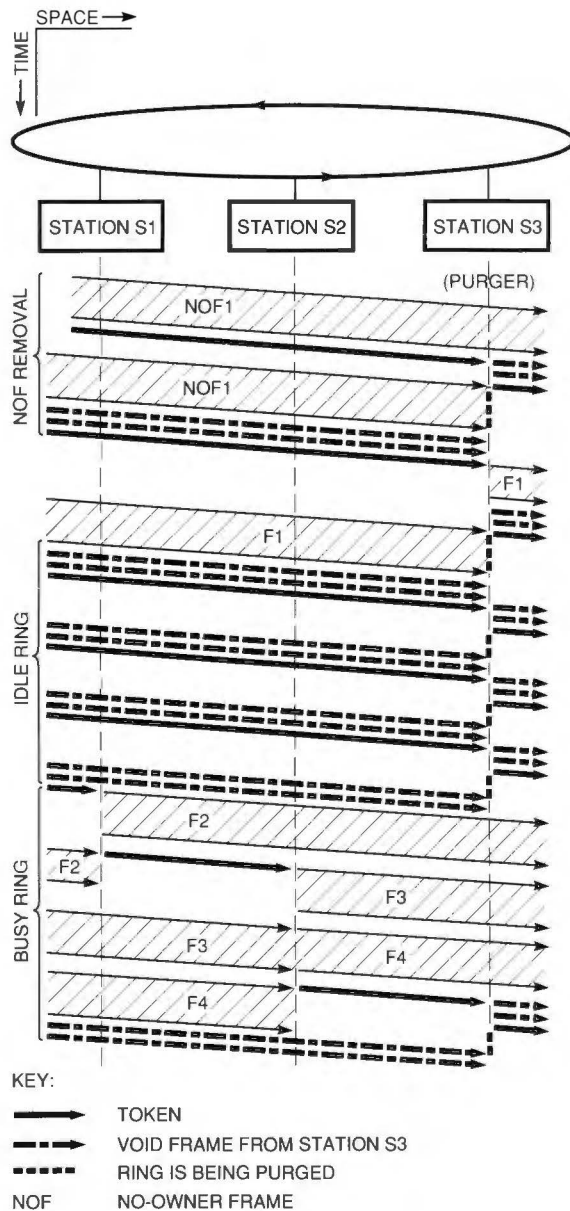


Figure 3 Ring Purging Example

In the worst case, the ring purger's effect on usable bandwidth is less than 0.22 percent. For implementations compliant to the ANSI FDDI MAC standard, these void frames have no effect since the standard prohibits copying void frames.

The ring purging algorithm removes NOFs, including long fragments, without disrupting the operation of the ring, and it removes NOFs within two traversals of the frame. In addition, it has an important property that permits more than one purger to operate in the same ring at any time. This property

allows the purger election algorithm to be more optimistic (i.e., when in doubt during election, one can start purging) during the transition period when the distributed election algorithm is stabilizing. The purger election algorithm is implemented in the Common Node Software (CNS), and the purging algorithm is implemented in the MAC chip.

FDDI Data Link Chips

The FDDI MAC sublayer functions are implemented by the three FDDI data link chips: the ring memory controller (RMC), media access control (MAC), and content addressable memory (CAM). The RMC interfaces between the frame buffer memory on the system side and the MAC chip on the network side. It consists of a direct memory access (DMA) engine designed to supply the MAC chip with frames to send and to store frames received from the MAC chip. The interface on the system side provides gathered reads on transmit and scattered writes on receive. Although the RMC's interface to the MAC chip was custom designed for FDDI operation, the RMC can, in principle, be used for other data links that run at 100 megabits per second or less. The MAC and CAM chips implement the FDDI MAC protocol functions. The functions implemented by the MAC chip include the token access protocols, frame delineation, frame parsing, address recognition, frame check sequence generation and verification, frame insertion, frame repetition, frame removal, token generation, and error detection and recovery algorithms (e.g., the beacon and claim algorithms). The CAM chip provides the destination address filtering, which determines if a received frame is to be received or discarded, and the setting of the A-indicator, which is part of the frame status field.

RMC Chip

The RMC chip is a high-performance coprocessor intended for full-duplex data transfer between the buffer memory and the MAC chip. It uses a pair of circular buffer queues for transmit and receive to manage DMA data transfers to and from the buffer memory. Two independent on-chip, first in, first out (FIFO) buffers, for receive and transmit, are provided to decouple the buffer memory from the real-time nature of the MAC interface. A fragment and frame filter is provided to reduce unnecessary memory accesses caused by the reception of fragments or frames not addressed to this station. As shown in Figure 4, the RMC chip has three interfaces: the processor interface, the MAC chip interface, and the buffer memory interface. The processor interface

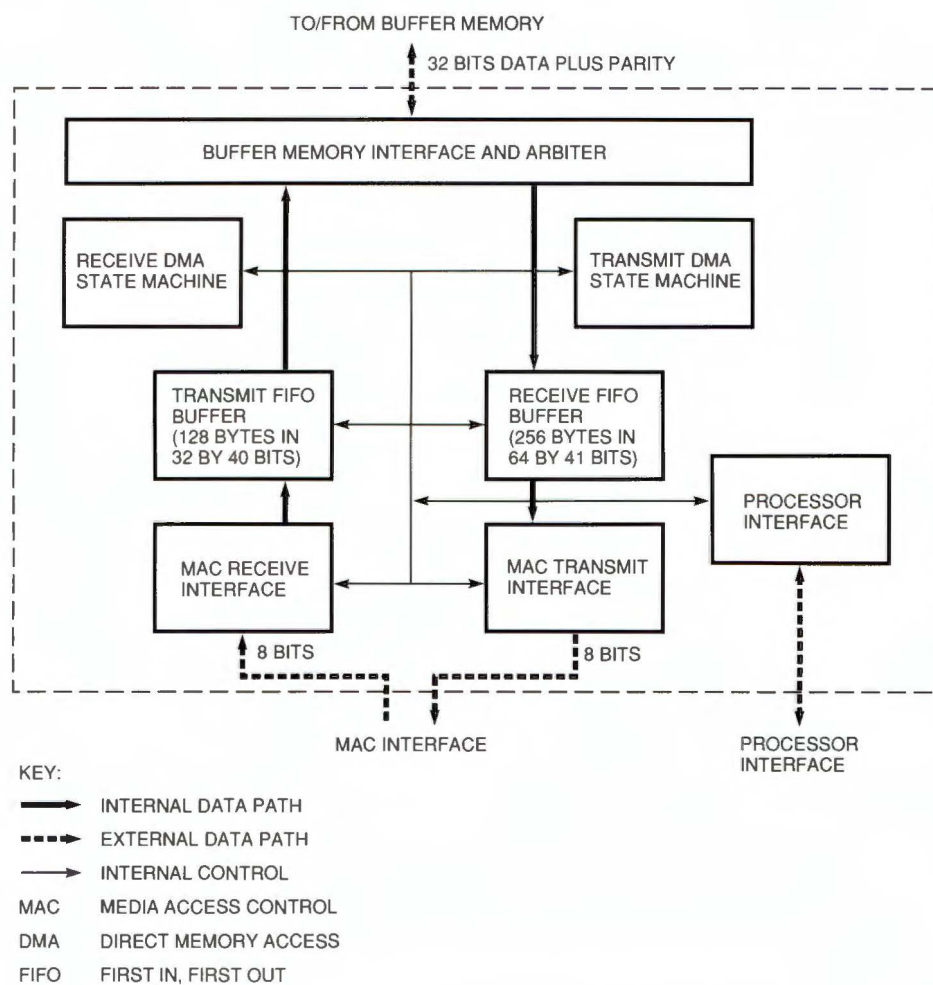


Figure 4 Ring Memory Controller Chip Block Diagram

allows the initialization, control, and observation of the RMC. The MAC chip interface consists of high-speed transmit and receive data paths for transfer of data and control information between the MAC chip and the RMC's FIFO buffers. The buffer memory interface provides a burst mode DMA for read/write from/to the buffer memory on a single bus arbitration cycle, where the burst size can be up to four or eight longwords. The RMC can provide a maximum data transfer rate of about 44 megabytes per second.

The RMC is implemented using a 1.5-micron-drawn, two-metal-layer custom complementary metal oxide semiconductor (CMOS) technology. It uses roughly 87,000 transistors, a large number of which are used for the two FIFO buffers, where the receive FIFO buffer is 256 bytes and the transmit FIFO buffer is 128 bytes. The RMC uses 102 signal pins and is available in a 132-pin cerquad package.

It is also a fully synchronous design [some self-timed logic is used in the FIFO random access memory (RAM) devices] using a 12.5-megahertz (MHz) primary clock and 25-MHz clock for sampling incoming signals. The FIFO RAM was implemented using a full custom methodology, and the remainder was implemented using an automated standard cell methodology.

The RMC interfaces to buffer memory using a data and address multiplexed bus, which is a 32-bit-wide bus plus the four parity signals and additional control signals. A bus transaction consists of an address cycle driven by the RMC followed by a burst of data cycles, either driven by the RMC on receive or driven by the buffer memory on transmit. One of the unique features of this chip is that it is able to use a 32-bit-wide buffer memory composed of low-cost [100-nanosecond (ns) access time] dynamic random-access memory (DRAM) chips, whereas many of the

other FDDI memory controllers available on the market require a 64-bit buffer memory or require very fast DRAM or static random access memory (SRAM) chips. To achieve these reduced memory and chip requirements, the RMC's buffer memory accesses were done in bursts of between one and eight longwords. Then, by making use of the DRAM's fast page mode, in which subsequent, sequential reads/writes are faster than the first one, the needed buffer memory bandwidth is attainable.

The RMC directly accesses a transmit ring and a receive ring, each consisting of a circular queue of descriptors. Each descriptor supplies the buffer memory address of a transmit buffer or a receive buffer. An OWN bit mechanism is used in each descriptor to determine if the descriptor and its buffer is owned by the RMC or not. It supports gathered read and scattered write in which frames to be received or transmitted can use one or multiple

buffers (and hence multiple descriptors), but a specific buffer/descriptor can only be used by one frame. Each receive buffer is required to be 512 bytes long. The RMC rewrites each descriptor in the receive ring to indicate the number of bytes actually used; and if the buffer is the last one for the frame, then the RMC writes the receive status and the frame byte count into the descriptor. Transmit buffers are also 512 bytes long and the RMC reads the size (in bytes) from the first descriptor for the frame.

MAC Chip

The MAC chip implements the FDDI MAC protocols, and it interfaces between the RMC or equivalent chip and the FDDI physical layer chip.¹³ As shown in Figure 5, the MAC has four interfaces: the processor interface, the RMC interface, the physical layer chip interface, and the CAM interface. The processor interface allows the initialization, control, and

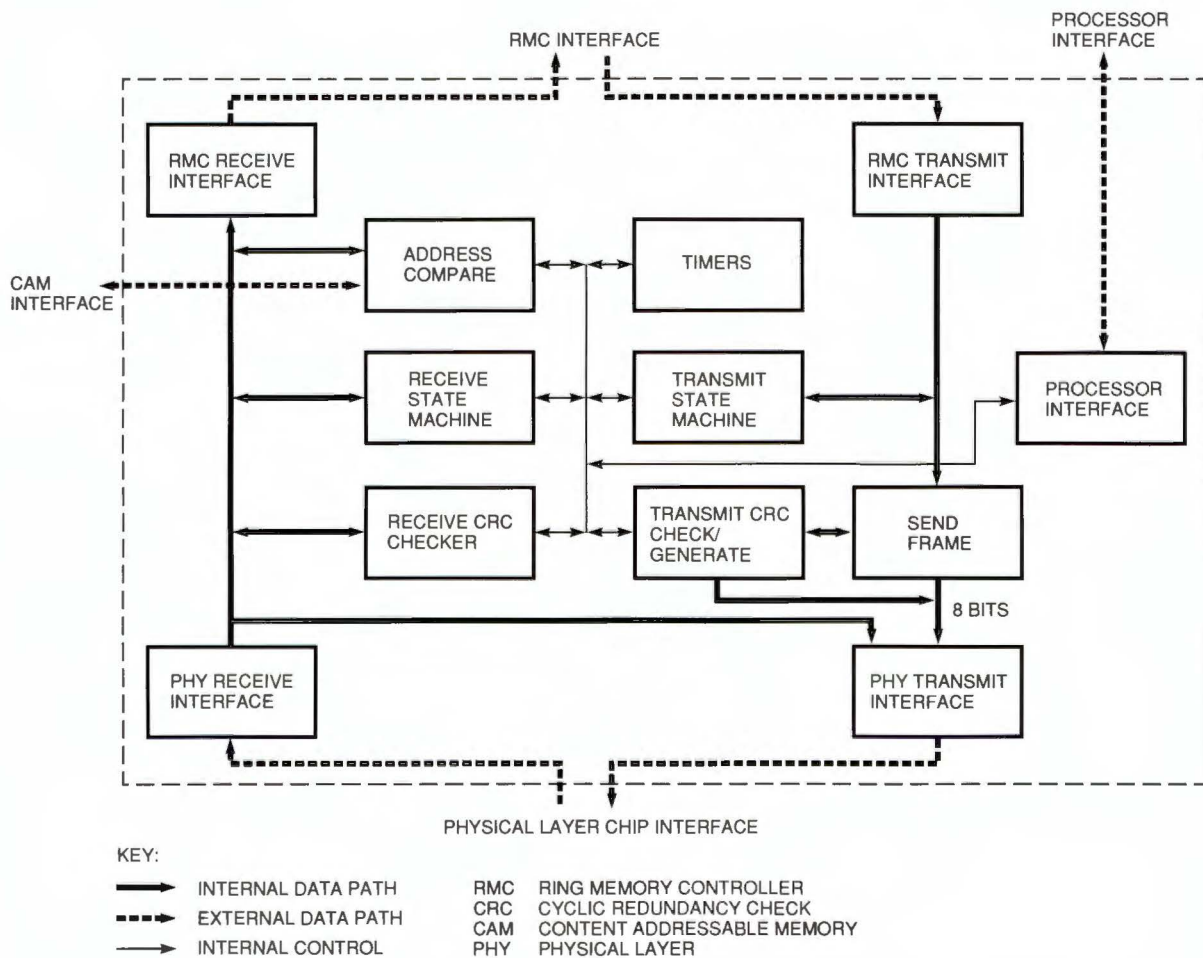


Figure 5 Media Access Control Chip Block Diagram

observation of the MAC. The RMC interface is custom designed for FDDI operation and allows the MAC to interface to either the RMC or to equivalent chips implementing the RMC interface. The physical layer chip interface allows the MAC to receive and transmit on the FDDI ring.

The MAC chip is implemented using a 1.5-micron-drawn, channelless, two-metal-layer CMOS gate-array technology and uses roughly 49,000 transistors (12,000 used gates). The MAC chip uses 86 signal pins and is available in either a 120-pin pin grid array (PGA) package or a 120-pin plastic quad flat pack (PQFP) package. This fully synchronous design uses primarily a single 12.5-MHz clock (80-ns cycle time); this operation is applied to the microprocessor bus as well. An additional double-speed clock is used in some of the peripheral interface logic to sample incoming signals and prevent hold time problems associated with clock skew between different chips.

The internal structure of the MAC chip has a full-duplex architecture. No logic is shared between the receive and transmit portions of the chip. Hence this chip can receive and transmit simultaneously for an indefinite period. This capability complies with the ANSI FDDI MAC standard that implementations be able to receive, parse, and validate certain frames (e.g., claim frames and beacon frames) even while transmitting. Two separate frame check sequence (FCS) checker/generators are required for transmit and receive functions. The MAC chip calculates the FCS, which is specified as a specific 32-bit cyclic redundancy check, eight bits at a time. A one-bit implementation is much smaller, but requires a 100-MHz clock. Even with such a clock, it is not easy to implement one bit at that speed. A byte-wide implementation requires considerably more exclusive OR (XOR) gates, but exploits more of the inherent parallelism of the algorithm and hence can be implemented using slower clock speed.

One of the important features provided by the MAC chip is the support of a 3-byte packet request header for transmission. The use of the packet request header construct allows simple, pipelined processing of the transmit descriptor along with the transmit data at high speed. This construct allows a software device driver to build a transmit descriptor to precede and identify the frame, which is then passed through the system bus and DMA data movers for delivery to the MAC chip. Every frame transmitted by the MAC chip must first contain the packet request header, which is used as the transmit descriptor and is not transmitted

as part of the frame. The packet request header is used to instruct the MAC chip on how and when to transmit the frame. For example, it instructs the MAC chip on whether to append the FCS to the frame or not, or on the type of token to use when transmitting the frame.

CAM Chip

The CAM chip provides a 64-entry content addressable memory where each entry is a 48-bit address. Typically, the entries in the CAM consist of multicast addresses used by upper layer protocols and the data link management protocols. The CAM is used to parse the destination address field of each frame received to decide whether the destination address received matches one of the entries in the CAM. The result of the address match is then provided in real time as input to the MAC chip, which decides whether to receive or to discard the frame. In the worst case, the MAC protocol requires that the destination-address-match decision be completed in less than 10 bytes of time (i.e., 800 ns), starting from the end of the destination address field.

The CAM chip is implemented using a 1.5-micron-drawn, two-metal-layer custom CMOS technology and uses roughly 44,000 transistors—34,000 of which are used for the core array of 64 words of 48 bits (plus a valid bit). The CAM chip uses 37 signal pins and is available in a 44-pin cerquad package. It also is a fully synchronous design using the same 12.5-MHz primary clock (plus the 25-MHz clock for sampling incoming signals).

As shown in Figure 6, the CAM consists of three interfaces: the processor interface, the MAC chip interface, and the physical layer chip interface. The physical layer chip interface consists of an 8-bit bus which transfers data bytes from the physical layer chip to the MAC chip. Every 80 ns, one byte is loaded into the appropriate position in the 6-byte internal compare register. Once the last byte of a 48-bit address has been clocked into the CAM chip's compare register, a match/no match indication is given to the MAC chip within 120 ns by the MAC chip interface.

The other interface to the CAM is the processor interface, which is used by the processor to load and change the CAM entries. The processor does not directly read or write the CAM array, but instead reads and writes (16 bits at a time) to three 16-bit data registers and one 16-bit command register. By clearing the appropriate bit in the command register, the processor requests a read, write, or compare from the CAM array. The arbiter ensures that the

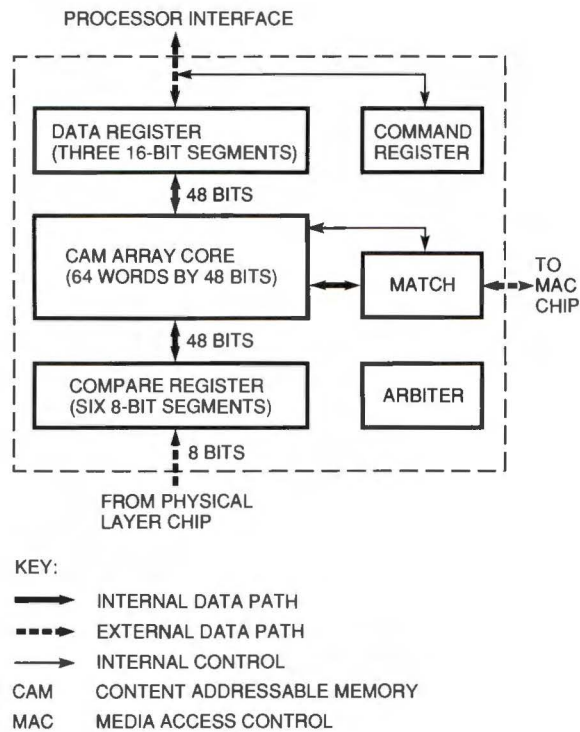


Figure 6 Content Addressable Memory Chip Block Diagram

CAM array is idle (i.e., not used by the compare operation) before allowing the processor's request to complete. In order to ensure atomicity for read, write, and compare, this arbitration is required because the actual CAM array can only perform one transfer or compare at a time. The arbitration mechanism guarantees that the compare register access is never delayed, since the MAC chip requires the match indication to be valid during a specific clock cycle. The CAM is also designed to allow entries to be added and removed at any time (even while the FDDI ring is running).

FDDI Simulation and Verification

Digital's FDDI technology development method was a top-down approach, starting with high-level system models of FDDI behavior and progressing to more detailed behavioral and structural models as confidence in functionality increased. Several simulation models and analytical models were developed to study and model the FDDI at the architecture and system levels. Using these models, studies were done on the error characteristics and robustness of FDDI, stability of the ring topology, performance and operational behaviors, and correctness of the protocols.¹

Every chip was partitioned into logical subblocks and a DECSIM (Digital's simulation tools and language) block-behavioral model was developed for each subblock. The external interfaces and internal structure of the model accurately represented some unit or subunit of a chip. When the model interfaced to some nonexistent model (e.g. unwritten behavioral model or external interface to another chip), a transactor was provided. The transactors provided accurate models of interfaces (timing, control, and data signals), but lacked the internal detail of a behavioral model. The use of the transactors was particularly important when modeling the buffer memory interface, since the design of the interface is implementation dependent (designed according to product needs by the development group using the FDDI chip set).

Each block-behavioral model was tested in isolation; then all were combined to produce a behavioral model of the target chip. When the chip model was successfully tested, each behavioral subblock was replaced by a corresponding structural model representing gate/transistor logic. The new chip model was then retested until the structural model behaved identically to its behavioral counterpart. As the model of each chip was completed, the transactors driving its external interfaces were replaced with the model for the next adjacent chip. The resulting combination was then tested together using the remaining transactors and test vectors. This process tested interoperability between chips and was repeated until all chips in the chip set had been tested together as a system.

After chip layout was completed, the structural models were enhanced to reflect the more accurate timing data. The test vectors were again applied using a checker model, which consisted of one block-behavioral model and one structural model of the same chip. The test vectors were applied to each model simultaneously, and external and internal signals of both models were compared for consistency. Any discrepancy between the monitored signals was thoroughly investigated and corrected as necessary.

For test and debug, we planned to develop a dedicated hardware tester to test the physical FDDI chips. Unfortunately bugs found by using such a tester occur too late in the process—the chips are already built. In order to meet our time-to-market goal, we needed to maximize activity in the simulation environment. Rather than waiting for the hardware, we decided to develop and apply as many of the tester-based tests as possible in the simulation environment.

Simulation Test Bed

The version of DECSIM we were using provided a C language interface capability. By writing the tests in C language and making use of a common but small environment-specific interface library, it was possible to simulate the system behavior using the chip models. By rewriting the interface library, the same tests were run unchanged on the tester and were used in regression testing of the chips during the fabrication process. The tests, interface library, and chip models became known as the simulation test bed. This was the first attempt to use this DECSIM capability as a cornerstone of a development strategy.

All the chip models were combined to construct an entity resembling a single attachment station (SAS) which was then tested as a full system, that is, a single operational FDDI node (in loopback). Extensive use of mixed mode simulation (mixing transactors and behavioral and structural models) aided test bed performance because the level of model detail could be varied, depending upon the area being tested. Time was saved by substituting higher level models in areas peripheral to those under test.

The simulation cluster was a cluster of four VAX 8840 systems, each system having four processors. Some idea of the extent of the effort expended can be conveyed by the following statistics:

- The total number of CPU hours used for the design and verification effort was 30,240 (196,560 VAX 11-780 CPU hours equivalent).
- For the single-node test bed, there were 827 MAC and 384 RMC tests. Using the 8840 cluster, the test suites required 336 (2,184 VAX 11-780 CPU hours) and 192 hours (1,248 VAX 11-780 CPU hours), respectively, for completion.

The individual tests varied in complexity from those requiring a few CPU minutes to those requiring days to run. For example, one MAC test which loops back ten 512-byte packets within the single-node test bed required 36 CPU hours to complete.

The importance of the test bed cannot be overstressed, as it is the major innovation in Digital's development methodology. Some CNS firmware was also developed in this environment. The benefits provided include:

1. The tests assisted the chip designers to discover bugs in the chip designs at the correct stage of development—in design rather than after silicon.

2. It was far easier to debug the tests in the simulation environment rather than on the physical hardware. During simulation we could observe and control chip behavior. Test results were easy to determine with clear pass or fail indications, and no human interpretation of large strings of 1's and 0's was required.
3. Since all test code was developed in C, no specialized test language was required, and standard support tools were readily available. Libraries of reusable (debugged) test-support functions were available, providing such functions as create frame, compare frame, write control/status register, and read control/status register.
4. Easy transition back and forth between the physical and simulation environments was important. Any bugs found in the physical environment had to be reproduced in the simulation environment in order to test bug fixes.

FDDI Tester

In parallel with chip design, another development group was assigned to test the chip set in a prototype FDDI system. Their approach was to design an FDDI tester that used the FDDI chip set. We also wanted to make this tester configurable as various FDDI entities, e.g., an SAS, a wiring concentrator, or an Ethernet-to-FDDI bridge, so that we could use the FDDI tester to build an FDDI ring to investigate the behavior of the ring.

The main value of the tester was its capability to perform long-term, steady-state testing, using billions of frames. It was also required to test complex topologies using multiple testers (as FDDI stations) in large rings. These activities are prohibited in the simulation environment because of the excessive amount of compute time required. The tester had to be capable of driving the data at full FDDI bandwidth, introducing controlled error conditions on the fiber, and accurately monitoring activity on the ring at full FDDI bandwidth. The tester itself was constructed so that it could be controlled via an external Ethernet link and multiple testers could be synchronized via external clock and control lines.

Testing the First Chips

The first pass of chips tested free of major defects. Most tests applied in the test bed passed the first time; the few exceptions were due to unrealistic timing expectations of the tester environment.

The next stage of testing involved combining testers into multinode FDDI configurations and

exchanging data at full speed over extended periods. This stage was also successful; no additional bugs were discovered, and the data link performance and ring stability exceeded the expectations. The steady-state testing was then augmented by introducing pathological conditions into the ring such as stations with duplicate addresses and noise on the fiber. The promiscuous capture modes and frame/event time stamping proved invaluable in analyzing subsequent behavior in these cases as the effects were often complex. Station management implementation was highly stressed and performed without error.

Finally, the same testers were configured as bridges, and prototype Ethernet-to-FDDI bridging firmware was introduced. At this stage several minor deficiencies with the control algorithms required for bridging were detected. These deficiencies were due mainly to insufficient analysis of FDDI bridging requirements by the test team; therefore, the test cases were correspondingly inadequate. All problems were repeatable on the simulation test bed, and bug fixes with new tests were developed. No deficiency was severe enough to prevent testing the prototype FDDI bridges and the development of more efficient algorithms for the FDDI bridge products.

Conclusion

The development of the FDDI data link and the chip set represents a major accomplishment and technical breakthrough in the high-speed LAN area. Significant contributions were made by Digital in the area of FDDI MAC algorithms and protocols to improve the performance and robustness of the FDDI LAN. The FDDI data link chips described in this paper are used in all members of Digital's FDDI product line, including bridges, wiring concentrators, and adapters. These products have benefited tremendously from the verification and test method adopted. Digital has built on its knowledge and experience in systems, networks, computer-aided design/simulation, and semiconductors to provide FDDI design, development, and methodology. Again, Digital has shown industry leadership by producing the FDDI chip set and products.

Acknowledgments

The authors acknowledge the technical contributions of Raj Jain, Brian Myrick, Chuck Lee, and K.K. Ramakrishnan. The authors would like to thank Jerry Hutchison, Bill Cronin, and Raj Jain for their review and several useful comments on this paper.

References

1. R. Jain, "Characteristics of Fiber Distributed Data Interface (FDDI)," *IEEE Transactions on Communications*, vol. 38, no. 8 (August 1990).
2. R. Jain, "Performance Analysis of FDDI Token Ring Networks: Entity Effect of Parameters and Guidelines for Setting TTRT," *Proceedings of the ACM SIGCOMM '90* (September 1990).
3. *Standard for the Transmission of IP Data-grams Over FDDI Networks*, Internet Engineering Task Force (IETF) RFC 1103 (June 1989).
4. M. J. Johnson, "Fairness of Channel Access for Non-Time-Critical Traffic Using the FDDI Token Ring Protocol," NASA Ames Research Center, Research Institute for Advance Computer Science, RIACS TR 86.9 (March 1986).
5. *Token Ring Access Method and Physical Layer Specifications*, ANSI/IEEE Standard 802.5-1989 (New York: The Institute of Electrical and Electronic Engineers, Inc., 1989).
6. R. M. Grow, "A Timed Token Protocol for Local Area Networks," *Electro/82, Token Access Protocols (17/3)* (May 1982).
7. P. Ciarfella, D. Benson, and D. Sawyer, "An Overview of the Common Node Software," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 42-52.
8. *MAC Bridge Specifications*, ANSI/IEEE Standard 802.1d (unapproved draft), P802.1d/D8 (New York: The Institute of Electrical and Electronic Engineers, Inc., 1989).
9. B. Stewart, W. Hawe, and A. Kirby, "Local Area Network Connection," *Telecommunications Magazine* (April 1984).
10. H. Yang and K. K. Ramakrishnan, "Frame Content Independent Stripping for Token Rings," *Proceedings of the ACM SIGCOMM '90* (September 1990).
11. K. K. Ramakrishnan and H. Yang, "Frame Content Independent Stripping for Token Rings," *Proceedings of the 15th Conference on Local Computer Networks* (October 1990).
12. *Fiber Distributed Data Interface (FDDI) Media Access Control*, American National Standard, ANSI X3.139-1987 (June 1988).
13. J. Hutchison, C. Baldwin, and B. Thompson, "Development of the FDDI Physical Layer," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 19-30.

An Overview of the Common Node Software

To address an aggressive fiber distributed data interface (FDDI) program schedule and reduce the complexity of the concurrent development of multiple FDDI products, Digital developed a common implementation of the FDDI station management standard. This implementation, called the Common Node Software, manages the physical and logical connections to the 100-megabit-per-second fiber-optic ring for Digital's FDDI product set. Including the Common Node Software in each product yields consistent behavior at the FDDI data link and physical layers. Direct reuse of the software reduces the development and testing efforts by providing a proven implementation.

The fiber distributed data interface (FDDI) data link provides clients with a connectionless data transmission and reception service. An essential element of this service is reliable connection to the physical network, allowing clients to transfer information across the network.

The Common Node Software (CNS) implements the part of the FDDI station that controls and monitors connections within the FDDI network. To provide this service, CNS implements the protocols defined by the FDDI station management (SMT) standard plus Digital's value-added enhancements and, in addition, manages the services provided by the FDDI chip set. These services include the ring memory controller (RMC), media access control (MAC), and elasticity buffer and link management (ELM) chips.^{1,2,3} CNS does not provide International Standards Organization (ISO) 8802-2 logical link control (LLC) support, which defines how data is reliably exchanged between two communicating systems. Although not provided by CNS, LLC support is included in each member of Digital's FDDI product set. Figure 1 illustrates the functional requirements fulfilled by the Common Node Software.

In this paper, we begin with a discussion of events leading to the development of CNS. Next, we present a detailed functional description of the FDDI station management and chip management services and specifics of the core and external libraries of the common code. We then describe the CNS development effort. A summary of the testing process follows, including details of the design verification test (DVT) monitor, developed by the

team, and interoperability testing among FDDI vendors. Finally, the benefits of common code realized by the project team are discussed.

Background of the CNS Development Effort

When development efforts for the DECbridge 500 and DECconcentrator 500 firmware began, the American National Standards Institute (ANSI) FDDI MAC, physical layer (PHY), and physical medium dependent layer (PMD) standards were complete, but the station management specification was not. The SMT draft was frequently changing; new protocols were defined and modifications to existing protocols were made with each meeting of the X3T9.5 SMT working group, which is responsible for developing the standard.

These changes complicated the concurrent development schedules of the DECbridge 500 and DECconcentrator 500 data link software. If two independent firmware teams designed their own FDDI data link software, both teams would need to follow the development of the SMT draft. These efforts could result in different interpretations of the SMT protocols, of how Digital's FDDI chip set works, and of the functions data link software should and should not perform. Producing multiple, independent SMT implementations could lead to incompatible products that exhibit inconsistent behavior and are unable to communicate.

During the evolution of the SMT draft, the physical connection management (PCM) protocol pseudo-code defined by the draft changed often; some

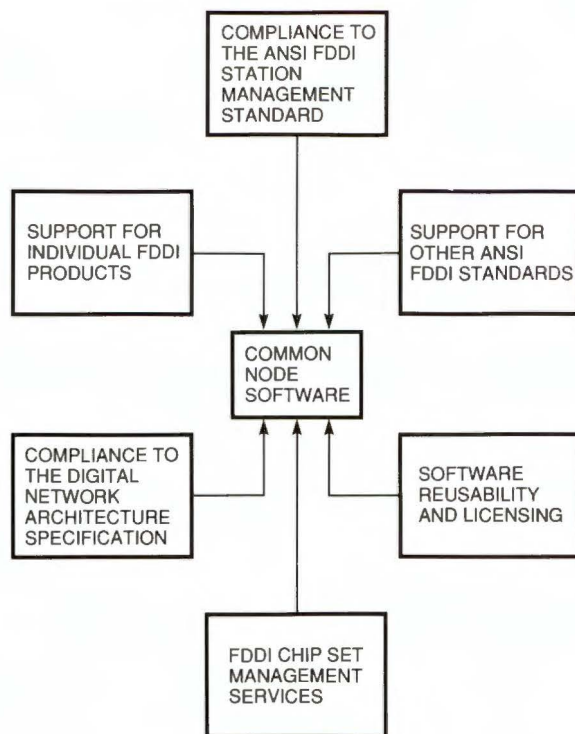


Figure 1 Functional Requirements Fulfilled by the Common Node Software

changes caused previous versions to be incompatible. PCM is responsible for the management of full-duplex physical connections between two FDDI PHY entities. The PCM pseudocode defines a synchronized bit-signaling communication process between two connecting nodes to exchange connection information. If two nodes attempting to connect to one another execute incompatible versions of the PCM pseudocode, these nodes will not connect.

To avoid this scenario, and possibly others, a decision was made to produce a common, reusable implementation of the SMT protocols and FDDI chip management. The initial goal of the CNS project team was to constrain the domain of possible SMT-related problems which could appear during the development of the FDDI product set.

Another important goal of the CNS project was compliance with both the SMT standard and the Digital Network Architecture (DNA) FDDI data link functional specification. Compliance with the SMT standard could increase the probability that Digital's FDDI products would interoperate with those of other vendors. Compliance with the DNA architecture would guarantee interoperability among Digital's current and future products.

The CNS team worked closely with Digital's representatives in the ANSI X3T9.5 FDDI SMT working group to develop the functional requirements of the software. As the SMT specification evolved, the functional requirements were completed and the development of the software began. The end result is a set of reusable software libraries which provide the functions necessary to implement the SMT protocols and to manage the FDDI chip set on each FDDI product. This reusable code, called the Common Node Software, is shared by the DECbridge 500 and DECconcentrator 500 products and also by more recently introduced FDDI products, such as the DECcontroller 700 adapter.

As the CNS design matured, the advantages and benefits of a common code implementation became more apparent to the FDDI program team. Originally, the design constrained CNS to operate under the same operating system used in the DECconcentrator 500 and DECbridge 500 products. CNS was extended to support microprocessor and operating system independence to accommodate future FDDI products, thus facilitating the portability of CNS to other environments.

This extended CNS support allows a variety of implementations, including host-based network controllers and firmware-resident drivers. Consequently, Digital is benefiting by distributing the software externally through third-party licenses to promote rapid introduction of quality FDDI products to the expanding marketplace.

Functional Description

This section presents details of the station management standard services and the management of services provided by the FDDI chip set. A functional block diagram of the Common Node Software is shown in Figure 2.

Station Management

The station management standard defines the services used by every station in an FDDI ring to monitor and control both the station itself and the state of the ring. The functionality defined by the standard includes connection management (CMT), ring management (RMT), and SMT frame-based services.

Connection Management CMT is primarily responsible for the maintenance of physical connections to the FDDI ring. This involves initializing and establishing connections between physical layer port (PHY port) entities and configuring the internal data path of a station. CMT is divided into the

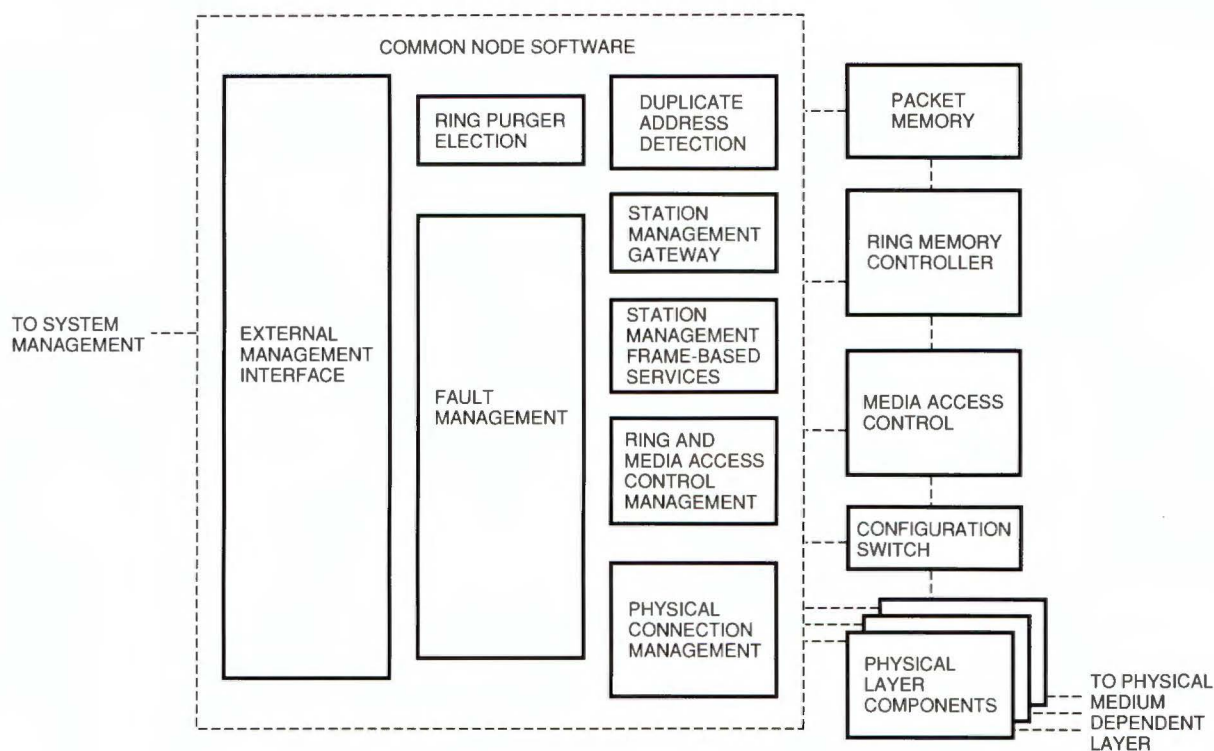


Figure 2 Functional Block Diagram of the Common Node Software

following three areas: entity coordination management, which coordinates the trace process and manages the optional optical bypass function; configuration management, which manages the configuration of the PHY and MAC entities within a station; and PCM, which manages the physical connection between a station's PHY and the PHY of the adjacent station. Table 1 lists the functions and divisions of CMT and summarizes the value added by Digital.

CMT provides a link confidence test and a link error monitor to check the quality of the physical link. When a connection is formed, the link confidence test invoked by PCM is performed to determine if the quality of the connection is adequate for proper ring operation. If the test fails, the connection is not allowed to form.

The link error monitor checks the quality of the connection after it forms by computing and monitoring the link error rate to determine if this rate is acceptable. If the rate falls outside the acceptable range, the connection will be terminated. A value-added feature of the link error monitor is error detection exceeding that required by the SMT standard.³

CMT is also responsible for topology control on new connections. Topology control, implemented by CNS as part of the PCM pseudocode processing, enforces a set of connection rules defined to prevent the formation of illegal ring topologies. The topology rules utilized by CNS are more strict than the default rules suggested by the SMT standard. Connections that are clearly misconfigurations are not allowed to form.

CMT also provides support for the trace function, which is a recovery mechanism for the loss of logical continuity of the ring. The trace function is initiated by the station downstream from the logical break and is propagated upstream toward the break. Stations that receive the trace notification leave the ring and perform a diagnostic fault test in an attempt to locate a faulty MAC or data path. The station causing the break should fail its test and not rejoin the ring. Another value-added function in CNS is the enhancement of the trace algorithms to ensure proper termination of a trace even in the presence of simultaneous network reconfiguration.

Connection management is implemented in both hardware and software. CNS provides the CMT services that must be implemented in software but

Table 1 Connection Management Functionality

Name	ANSI SMT Standard	Value Added	Description
Connection management	Yes		Manages physical connections and station configuration. Contains entity coordination management, configuration management, physical connection management, and link error monitor.
		Yes	Enhances topology management.
Entity coordination management	Yes		Coordinates the trace process and manages the optional optical bypass function.
		Yes	Enhances trace function to be insensitive to network reconfigurations.
Configuration management	Yes		Manages the configuration of the PHY and MAC entities within the station.
		Yes	Function is integrated within hardware.
Physical connection management	Yes		Manages the physical connection between connected PHY entities.
		Yes	PCM is implemented in hardware.
Link error monitor	Yes		Monitors active physical connection quality.
		Yes	Additional errors are recognized above those required by the standard.

are not performed by the ELM chip. For example, the PCM state machine, implemented in the ELM chip, specifies the timing, state, and physical bit-signaling used in the connection process. The PCM pseudocode, on the other hand, is implemented in CNS. This pseudocode is used to communicate connection information between neighboring PHY ports. Coordinating the PCM state machine and pseudocode provides full PCM functionality as defined in the SMT standard.

Another example of CMT services provided by CNS is the link error monitor. The ELM chip provides facilities to detect and signal the occurrence of bit errors on the link. CNS computes the link error rate based upon the data from the ELM and determines whether to sustain or to break marginal connections.

Ring Management RMT, also implemented in CNS, monitors the state of the logical link by using logical link status information received from the MAC sublayer. The information is then passed on to network management, such as Digital's extended LAN management software (DECelms), or to the LLC sublayer, for example, to report that the link is available for transmission service.

This MAC information is also used by RMT to detect and to resolve faults such as duplicate addresses and stuck beacon conditions. A stuck beacon condition occurs on the ring when a station's MAC continuously transmits MAC beacon

frames due to a fault condition on its data path. The logical ring does not form because the MAC is not able to receive its own beacons. Detecting the stuck beacon condition, RMT sends special MAC frames, called directed beacons, onto the ring to notify other stations of the condition. After sending these directed beacons for approximately 370 milliseconds, RMT tells CMT to initiate the trace function. When the trace function successfully completes, the data path fault is detected and the logical ring is formed.

If two or more stations have identical MAC addresses, the MAC protocols are adversely affected. If the ring becomes operational, these stations with duplicate addresses strip each other's frames from the ring, causing ring instability and increased packet loss. If the duplicate stations are performing ring initialization, however, successful ring initialization may be prevented and the ring will not become operational.

The RMT state machine can detect only the duplicate condition of its own MAC address. When this condition is detected, RMT has the option of removing the MAC from the ring, changing its address to a unique address, or forcing its MAC to lose the claim process and allow the ring to become operational. With the first option, a special control frame, called a jam beacon, is sent directly to the duplicate address to inform all stations with the duplicate address of the condition. LLC service

associated with the MAC is disabled while the duplicate condition is present.

RMT duplicate address detection is complemented by the neighbor notification protocol. This protocol allows a station to learn both its downstream and upstream neighbors' addresses. By transmitting a periodic neighborhood information request frame (NIF) to its downstream neighbor, the protocol is able to detect from the received NIF responses that its MAC address is duplicated on the ring. LLC service associated with the MAC is disabled until the condition is removed.

Station Management Frame-based Services Station management frame-based services include both mandatory and optional functionality. The mandatory functionality includes the NIF, status information frames, echo frames, request denied frames, extended service frames, status reporting frames, and the as yet undefined resource allocation frames. Parameter management frames are defined as optional functionality.

The set of SMT frame-based services supports higher-level network management functions which are not part of SMT. The information provided by the services helps network management to determine the topology and state of the ring and to control the network. The status report frame service announces status information to network management. The optional parameter management frame service facilitates remote management of FDDI stations. The status information frame service provides station configuration and operation parameters. The echo frame service provides station-to-station, loopback testing using SMT frames. Request denied frames are sent by a station in response to receiving service requests that are not understood or are not implemented. Extended service frames allow

the use of vendor-defined frames. This service provides the capability for exercising new SMT frame-based services.

These services are also useful for certain functions embedded within Digital's stations. These embedded functions utilize the SMT frame-based services and are summarized in Table 2.

CNS makes use of the extended frame service by implementing the ring purger election protocol, which supports the ring purger function. The ring purger is one of several functions defined by Digital's FDDI architecture that add value to the SMT standard.²

The ring purger election protocol is an algorithm implemented within CNS; the purging function is implemented in the MAC chip. This distributed algorithm elects one station on the FDDI ring to be the ring purger. Candidate ring purgers, using the SMT extended frame service, send ring purger election frames to a multicast address known only to stations participating in the algorithm in order to communicate with each other. The station that either wins the claim process or has the highest address becomes the ring purger. Once elected, a ring purger enables the purging in the MAC chip and sends out periodic "hello" messages to the ring. If these messages are not received after a period of time, the election process is repeated. Other error control and recovery procedures are built into the algorithm to increase robustness.

Another value-added feature of CNS is the SMT gateway protocol. This protocol helps build ring maps on network management consoles. A ring map is a database that can be used to build a graphical representation of the network topology. The map not only presents a visual image of the network, but also displays characteristics about each node on the ring such as its type and number of ports.

Table 2 Embedded Functions Supported by SMT Frame-based Services

Name	ANSI SMT Standard	Value Added	Description
Duplicate address test	Yes		Periodic NIF requests inform neighbor of station's existence, get neighbor's address, and test for duplicate of this station.
		Yes	Sending another NIF request to our own address improves coverage over ANSI required test.
Ring purger election	No	Yes	Digital's ring purger rids the ring of no-owner frames and fragments. Ring purger election is controlled by a distributed algorithm.
SMT gateway	No	Yes	Management may use any station as an agent to query other stations for SMT information.

The SMT gateway within CNS uses status information and neighbor information request frames to solicit information about other stations on the network. Responses received include information about a station's configuration, its network address, current counter values, and FDDI timer values. The gateway collects these responses and returns them, using the management protocol within the product, to the host management station building the map.

By providing a well-defined and protocol-independent interface to its clients, the SMT gateway can be used with any network management protocol. The first product to utilize the SMT gateway is DECelms software.¹

FDDI Chip Management

CNS manages the services provided by the FDDI chip set, including the initialization of all configurable chip parameters such as timer values and interrupt masks. Special chip modes, such as various frame reception modes supplied by the MAC, multiple loopback modes in the ELM, and parity detection in the RMC, are also controlled by CNS. Software control of these modes is supplied by a set of interface functions within CNS. Many of the SMT-related protocols, such as CMT and the ring purger algorithm, are implemented in both the FDDI chips and CNS. CNS controls the operation of these algorithms through its management of the chip set.

CNS also provides consistent FDDI chip fault management for all products using CNS. Compile time and system initialization options allow the product designers to specify, for each fault, whether it should be considered fatal or nonfatal to the system. Also, for each event, a threshold is set indicating the maximum number of times the event is to occur over a predefined period before any action is taken. When the threshold is reached and a fault is classified as fatal, CNS removes the station from the ring and notifies the firmware kernel. The kernel is then responsible for further action, such as rebooting the hardware or running a diagnostic test. If a fault is considered nonfatal, CNS notifies the kernel of the event, but the station does not exit from the ring.

In addition, the algorithm can disable a recurring event for up to one second. This provides flow control of fault events and allows other processing to continue. During lab testing of the DECconcentrator 500 prototype, broadcast packet storms caused receiver overrun conditions in the RMC memory controller. An interrupt event is associated with the overrun, so the event occurred continuously, using up all available processor time. This situation

effectively disabled the product until the storm ended. The flow control method was devised to throttle events in such a situation. Once the occurrence rate of the event slows down, the throttling stops.

Implementation Specifics

When the CNS development began, some important questions arose. Many of the questions were about FDDI itself. Most of the questions dealt with issues concerning the structure of the reusable software, for example,

- What defines common code?
- What functionality would be common across all products?
- How do we handle common functionality that must be implemented on different hardware platforms with different interfaces?
- What requirements, such as consistent interfaces, need to be placed upon external hardware and software?

Common Code

CNS is code written entirely in the C language, which can be executed on different hardware platforms without modifying the source code. This common code includes a library of core functions, which are completely portable, and a set of external functions to provide services that cannot be implemented in the same way on all products. The CNS core and external library interfaces with the remaining system firmware are shown in Figure 3.

The CNS project team developed a coding standard to facilitate a high degree of portability. The standard provides, for example, a portable set of type definitions to ensure that a long integer is always 32-bits wide and a short integer is always 16-bits wide. Other type definitions specify the size of control/status registers, which may be either 16 or 32 bits, depending upon the platform.

But implementations of unions and bit-field definitions among C compilers is inconsistent. The language allows the assignment of variable names to individual or strings of bits. Control/status register bit manipulation is easy to perform using the combination of structure and bit-field definitions. But the ordering of bit assignments can change from compiler to compiler. While one compiler assigns bit 0 to a bit-field declaration, another may assign bit 31. Correctly mapping bit definitions in software to their corresponding definitions in a hardware

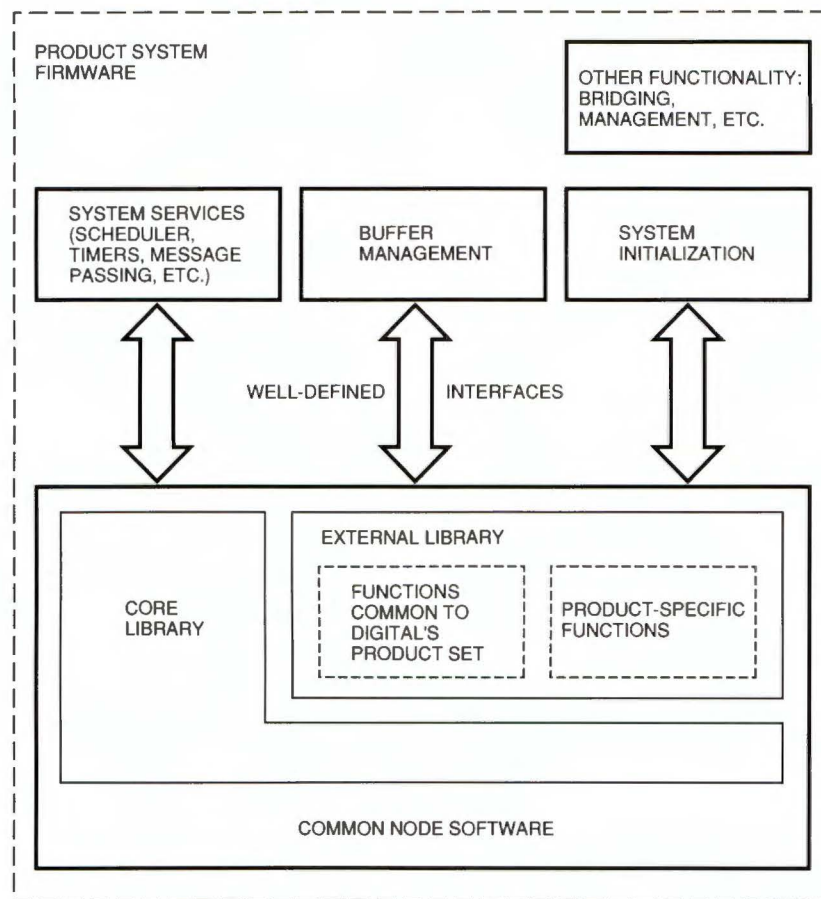


Figure 3 Common Node Software Interfaces

register cannot be guaranteed. Thus, the use of unions and bit-field definitions was eliminated.

The Core Library

The software is partitioned into two libraries containing core and external functions. The core library consists of a set of completely reusable functions. No source code changes are necessary to execute this software on different products and hardware platforms.

The core library provides the major functions of CNS, but relies on the external library to provide operating system and hardware support. The core provides a set of interface functions that manage the physical and logical connections to the FDDI data link.

The core implements many of the SMT protocols such as any CMT not performed by hardware, duplicate address detection, the SMT frame protocols, and Digital's ring purger election protocol. The core

library also provides interrupt processing functions for the MAC and ELM chips. The external library provides the interrupt service routines (ISR), and these core functions are called from within the ISR. Built within these interrupt processing functions are facilities to manage chip faults, such as parity errors or overrun conditions signaled by the FDDI chips.

The External Library

The external library is based on a well-defined set of functions such as allocating a transmit buffer, starting a software timer, or enabling the FDDI data path scrub function used to clear frame fragments from the ring. This library provides the direct interface to the product's operating system, buffer management services, and hardware configuration.

Digital's implementation of the external library is further divided into two sublibraries. The first sublibrary consists of functions specific, but common,

to Digital's FDDI product set and is completely portable throughout the set. The other sublibrary of functions is specific to each FDDI product and is not portable among them. These functions deal mainly with special hardware access and configuration management, such as inserting the MAC onto the physical data path internal to the DECconcentrator 500 product. The interfaces to these functions are the same for all products, but different hardware designs or the limited functionality in a product may require distinct implementations. For example, the DECconcentrator 500 device has an internal data path that interconnects the MAC and PHY entities within the product. Logic surrounding the MAC chip can be used to insert and remove the MAC from the data path or to bypass the MAC. The external library for the DECconcentrator 500 firmware contains two functions to insert and bypass the MAC. In contrast, the DECbridge 500 product's MAC is always on the data path and does not need to bypass or insert the MAC; therefore, the firmware does not contain bypass or insertion functions.

To facilitate consistent memory access, low-level packet memory functions enforce network byte order in SMT frames. Network byte order defines the order in which bytes are transmitted and received. These functions perform 16- and 32-bit read and write operations when building and parsing frames. Other product-specific functions provide access to designated packet memory locations for generating special MAC-level control frames called directed beacons, which are used by ring management in special fault situations.

Data Structures

Supporting both libraries is a set of three data structures, Phy, Link, and Station, that store state, configuration, and counter attributes information pertaining to CNS. The Phy data structure contains information about a single PHY port. The Link data structure reflects the state of the logical link associated with the MAC, while the Station data structure maintains information about the general state of CNS.

All information about the state of CNS and SMT is contained within these data structures. Other firmware agents residing within the products, such as extended LAN management software responders, need only to look in a central location for management information pertaining to SMT. Global variables defined by CNS for its sole use are kept to a minimum.

The structures are linked to reflect the actual configuration and data path of the MAC and PHY entities within the station. This linking provides easy

support for various configurations and for the execution of configuration-based protocols. For example, in the DECbridge 500 product, the CNS data structures consist of one Station, one Link, and one Phy connected to reflect the single attachment station (SAS) architecture. The DECconcentrator 500 firmware has one Station, one Link, and 12 Phy data structures connected to form the configuration of the station. A six-port concentrator with a management board is represented by one Station, one Link, and six Phy data structures. An eight-port concentrator without a management board does not contain a MAC, and, thus, is configured as one Station and eight Phy data structures.

The CNS Development Effort

The actual design and development cycle for CNS covered a six-month period. During this time the SMT draft standard went through many revisions; the concept of licensing the CNS code was introduced; and Digital's FDDI chip set underwent a second-pass design cycle.

The licensing effort required rethinking and making adjustments to the partitioning and structure of the design to accommodate several layers of interfaces and support functions in CNS. These requirements resulted in an additional four to six weeks of effort, but yielded the benefits of a code that could be used outside Digital and a more generic design to accommodate future product designs.

Of all Digital's FDDI chips undergoing second-pass design, the ELM chip required the greatest number of changes. The majority of these changes were due to the major redefinition of the physical connection management portion of the ANSI SMT draft.³ To minimize risk to the chip development effort, simulation of the connection management portion of the CNS code was performed on Digital's logic simulation system (DECSIM) test bed. This testing utilized one of the same test beds constructed by the ELM chip designers configured with two ELMs connected together.² This test bed utilized the behavioral chip models in order to speed execution. Additional routines were written to emulate resources normally provided by the operating system (e.g., timer services). Roughly one week was spent in simulation, two days of which focused on developing and debugging the environment. Testing included the initialization of a good connection as well as connections that resulted in topology rejects, link confidence test failure, and link error monitor failure. No bugs were found in the ELM design, however several coding bugs were discovered.

Later, when the first DECconcentrator 500 hardware was available in the lab, and the operating system services were debugged and available, CNS PHY code required only one-half day to debug before becoming operational. Thus, prior simulation of the code was clearly beneficial.

The first product that utilized CNS was the DECconcentrator 500 firmware. The integration, debug, and design verification process spanned an eight-week period. For subsequent products, the duration of this phase was greatly reduced: two weeks for the DECbridge 500 firmware and one week for the DECcontroller 700 adapter. This reduction in time was, of course, due to the reuse of the core and external libraries. Only a small subset of CNS was unique for each product; hence, debug time was minimal. Also, as experience was gained in verification testing, and the related tools improved, the test process became more efficient.

Testing the Common Node Software

One fortuitous advantage to the structure of CNS is that the core functionality only needs to be tested exhaustively on a single platform. The partitioning of core and external functionality and the external requirements to which each product environment must adhere yield this advantage. Thus, the only testing necessary on an individual product is the initialization of CNS and the external interface between CNS and the system firmware. These product dependencies include SMT frame transmission and reception, status and error message passing, and some functions unique to the product.

The task of test and verification of the CNS core presented some interesting challenges. The complexity and distributed nature of many of the algorithms made testing difficult. The complexity causes automation of the test process to be an extremely involved task. Some functions, such as RMT, can be tested only in a multiple-station configuration due to the distributed nature of the algorithms.

Another difficulty in performing the initial testing of CNS was the lack of visibility into the executing software. The DECelms product reports some information maintained by CNS, but most of the data used during testing is not visible to network management. Also, the DECelms product was being developed at the same time as CNS and was not ready for use. In addition, there was no global visibility into the ring. At the time, commercially available FDDI datascopes or analyzers that would have been used to view symbol streams on the fiber were not available.

To facilitate testing, the CNS team developed a tool referred to as the design verification test (DVT) monitor. This tool provides a detailed view into the operation of CNS as well as automated tests for CNS interface functions. The tool also has the capability to insert faults into the ring and exercise many of the SMT protocols.

The DVT monitor has two components: a connection to a universal asynchronous receiver/transmitter (UART), which provides serial communication between the product and a display terminal; and monitor software, which is layered on top of CNS. Thus, the monitoring and managing of CNS is achieved by out-of-band access via the UART. This access is necessary to perform testing of ring fault conditions. The monitor software is run at a lower priority than all other system components to leave the system timing or operation unaffected. This software provides both nonintrusive or "peek" and invasive or "poke" management capabilities. Password protection on a login screen prevents unauthorized users from disrupting the network. To use the tool, one must log on to an FDDI product running the test software.

As a nonintrusive tool, the DVT monitor provides passive monitoring of the network status and related events. The tool provides real-time monitoring of all physical (or port) and logical (or MAC) connections in the product. Status windows continuously display the state of all physical and logical connections to the ring.

As an invasive tool, the monitor can be used to insert faults and to exercise the ring. The tool can be used to easily change station parameters, such as the station's address. Configuration hardware within each product can also be changed to affect the operational state of the ring. Changing this hardware is especially helpful for introducing duplicate addressed stations, stuck beacon conditions, beacon/claim ring oscillations, and other anomalies into an operational ring in order to exercise RMT and the trace function. To analyze SMT frames, an SMT frame agent exists in the tool to generate and receive any SMT frame type, including frames not defined by the standard.

Prior to the development of the DVT monitor, the FDDI tester was utilized to generate and receive SMT frames to test the proper operation of the SMT frame-based protocols.² In later testing, the FDDI tester was indispensable in creating a variety of traffic loads on the ring to test the products' responses to traffic loads. The tester allowed an arbitrary mix of frames sent at a programmable rate over the

FDDI ring and made it simple to characterize the responses of the products to a wide range of network traffic conditions.

FDDI Interoperability Testing

Interoperability testing among the FDDI vendors is important for many reasons. Comprehensive interoperability testing among the vendors both reinforces the correct interpretations of the standard and performs the more immediate goal of verifying the correctness of the implementations. The proper operation of all stations on the network both in normal operation and in response to fault conditions is necessary if the commercial marketplace is to fully accept FDDI.

The nature of a ring topology requires the active participation of each station up to the MAC sublayer. Each FDDI station must establish and maintain physical connections and repeat frames without error. If a single station does not repeat frames, ring connectivity is lost. The X3T9.5 standards committee anticipated faults that prevent normal ring operation and devised algorithms to deterministically resolve such conditions. The addition of fault recovery schemes devised for the FDDI system, while necessary to guarantee proper operation of the network, made the standard more complex. Correct implementation of these complex protocols and distributed algorithms is essential to ensure that one vendor's implementation will operate correctly on the same ring with other vendors' implementations—especially in the presence of fault conditions.

Every vendor must make certain that its implementation adheres to the functionality as defined by the SMT standard. Digital encouraged cooperative testing among the vendors and participated in testing with many vendors at Digital's FDDI development center in Littleton, Massachusetts, at customer sites, and at other vendors' locations.

Interoperability Test Methodology

A test plan was developed originally for design verification and, later, for interoperability testing between Digital's FDDI product set and products from other vendors. The test plan concentrates mainly on the interoperability of the FDDI data link, defined by the FDDI PHY, MAC, and proposed PMD and SMT standards.

The plan covers connection management, ring management, and SMT frame-based services and functions defined in the SMT draft standard. The intent of the plan is to verify plug-and-play capability as

well as to correct operation under both normal and aberrant network conditions.

Connection management testing covers the physical connection management and configuration management processes. PCM testing covers the bit-signaling and connection initialization algorithms, the link confidence test and link error monitor, and verification of the connection matrix defined in the SMT draft. CFM testing verifies the correct operation of the reconfiguration scrub and MAC insertion functions.

Ring management testing covers duplicate address detection, including stuck beacon detection and recovery, directed and jam beacon initiation and reception, and the trace function. Other miscellaneous testing monitors the abusive use of restricted tokens and extended service frames.

Frame-based testing covers all required SMT frame protocols. These protocols are tested extensively for compliance to the SMT draft. Parameters within the frames are examined for consistency and correctness. For example, all timer values presented in SMT frames are verified to be in two's complement form, and all canonical addresses are correctly converted to FDDI most significant bit order.

Results of Interoperability Testing

The interoperability testing uncovered problems in many vendors' implementations, including Digital's.⁵ Many of these problems resulted from inconsistent interpretations of the SMT draft; others were due to incomplete implementations that did not support some functions defined in the SMT draft; and still other problems could be attributed to changes in the SMT draft overlooked by some implementations. As a result of the testing, many of these problems have been fixed, and the number of interoperability problems within FDDI networks has been reduced.

Conclusion

The Common Node Software provides the FDDI project with a very flexible and stable implementation of a major portion of the FDDI data link. The initial investment in time spent on development was longer than that expected for independent software development efforts but is justified by the long-term benefits of common code. Independent development efforts for the DECconcentrator 500 and DECbridge 500 products, for example, probably would have taken less time. Each design would be based upon the hardware design and system requirements of each product. Independent designs

would reduce the amount of software developed for each product because the designers would not have to address portability issues, such as generic interfaces to the operating system or hardware.

But these two products were the first of a new network architecture, and the station management draft was not complete during product development. Thus, independent implementations would have resulted in a higher bug rate, with each product exhibiting its own set of behavior at the SMT level. Future updates and revisions to the SMT standard would have resulted in independent revisions of each product's firmware and, possibly, a new set of problems. With CNS, only one source needs to be changed and tested.

The FDDI standard promotes multivendor interoperability, so that independent products can communicate effectively in a heterogeneous FDDI network. The development of CNS significantly increased interoperability between Digital's products and those of other vendors.

The advantages of reusable software that were realized by the project team can be summarized as follows:

- The major design of the software needs to be done only once. A core library and a well-defined external interface are provided. When a new product is developed, only the specific implementation of the external interface needs to be supplied.
- Test and verification time is significantly reduced. Only the interfaces to CNS and the system dependencies must be rigorously tested with the design of each new product.
- The bug rate is reduced significantly. Each new product uses the pretested and proven core library.
- The software requires little maintenance. Since the core library is stable, development only needs to be performed on the external library.

The CNS development project was the design team's introduction to reusable software. We have probably not done everything in the best possible way, but the success of the project and the time and effort saved have convinced us of the benefits of reusability.

Acknowledgments

The authors would like to thank Peter Hayden and Herman Levenson for their technical contributions to the development of CNS. Much of the structure

of the common code is a product of their ideas. Appreciation is also extended to Bill Cronin, who was instrumental in expanding the scope and effectiveness of the interoperability testing. Finally, the authors would like to acknowledge Henry Yang for his careful and constructive review of the paper.

References

1. *FDDI Station Management*, Draft Proposed American National Standard, X3T9.5/94-89, REV 6.2 (May 18, 1990).
2. H. Yang, B. Spinney, and S. Towning, "FDDI Data Link Development," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 31-41.
3. J. Hutchison, C. Baldwin, and B. Thompson, "Development of the FDDI Physical Layer," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 19-30.
4. B. Sweet, "DECelms—Managing Digital's FDDI and Ethernet Extended Local Area Networks," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 76-84.
5. D. Benson, P. Ciarfella, and P. Hayden, "FDDI—Meeting the Interoperability Challenge," *Proceedings of the IEEE Fifteenth Conference on Local Computer Networks* (October 1990).

Development of the DECbridge 500 Product

The DECbridge 500 product connects Ethernet/802.3 local area networks (LANs) to fiber distributed data interface (FDDI) LANs and is, therefore, a fundamental element of an extended LAN. Developers of this product encountered many technical hurdles. The higher data rate and token ring topology inherent in the FDDI technology impose several demands on any bridging product connected to an FDDI LAN. The differences in formats and size of frames on the two types of LANs introduce further requirements. The development team met these requirements and delivered a high-performance product that provides seamless integration of both LAN types.

Bridges are essential to the creation of extended local area networks (LANs) because they provide transparent forwarding of traffic between adjacent LANs.¹ Traffic may be forwarded to or from individual destinations, to groups of destinations (multicast), or to all destinations (broadcast). Bridges only forward traffic destined for other LANs; local traffic is confined to its home LAN.

One important function of bridges is the ability, under network management control, to block traffic of selected protocol types or traffic from specific sources. Restricting unnecessary traffic, especially multicast or broadcast, significantly improves the utilization of LAN bandwidth.

In this paper we first discuss the role of the DECbridge 500 product in an FDDI and Ethernet/802.3 extended LAN and outline the design of the bridge. We then describe the operation of the bridge by tracing the flow of LAN traffic through it. This description gives insight into many of the complex tasks that a bridge must perform to connect two dissimilar LANs. Key points of the development methodology are also presented.

DECbridge 500 Design Considerations

The DECbridge 500 device serves as the point of connection between a new family of LAN products based on the fiber distributed data interface (FDDI) technology and a large installed base of Ethernet/802.3 LANs. The DECbridge 500 product must meet the requirements of both LANs to provide a smooth migration path for Digital's customers. Note that Ethernet and 802.3 have media access control (MAC) frame formats that may be used on the same 10-megabit (Mb)-per-second LAN.

Throughout this paper, the expression Ethernet/802.3 is used to identify such LANs and to distinguish them from 100-Mb-per-second FDDI LANs. The terms Ethernet, 802.3, and FDDI are used when discussing the specific MAC frame formats.

System Description

Two typical extended LAN applications involving DECbridge 500 devices are shown in Figure 1. The backbone application employs an FDDI LAN to provide a high-bandwidth interconnect of multiple Ethernet/802.3 LANs. The DECbridge 500 device is the point of connection between the Ethernet/802.3 LAN and the FDDI backbone LAN. In the work group application, FDDI LANs provide localized connectivity of users, such as DECstation 5000 workstations, that have high throughput requirements. File servers and other common resources may also be part of the local FDDI LAN. Here, the role of the DECbridge 500 product is to provide a path from the local work group to other parts of the extended LAN via Ethernet/802.3 LANs.

In either application, the bridge must perform the following functions:

- Forward traffic between nodes residing on two different LANs
- Prevent (i.e., filter or not forward) traffic between nodes on the same side of the bridge from getting to the LAN on the other side of the bridge
- Be responsive to host-based network management, provided by, for example, Digital's extended LAN management software (DECelms) and Digital's management control center (DECMcc) product

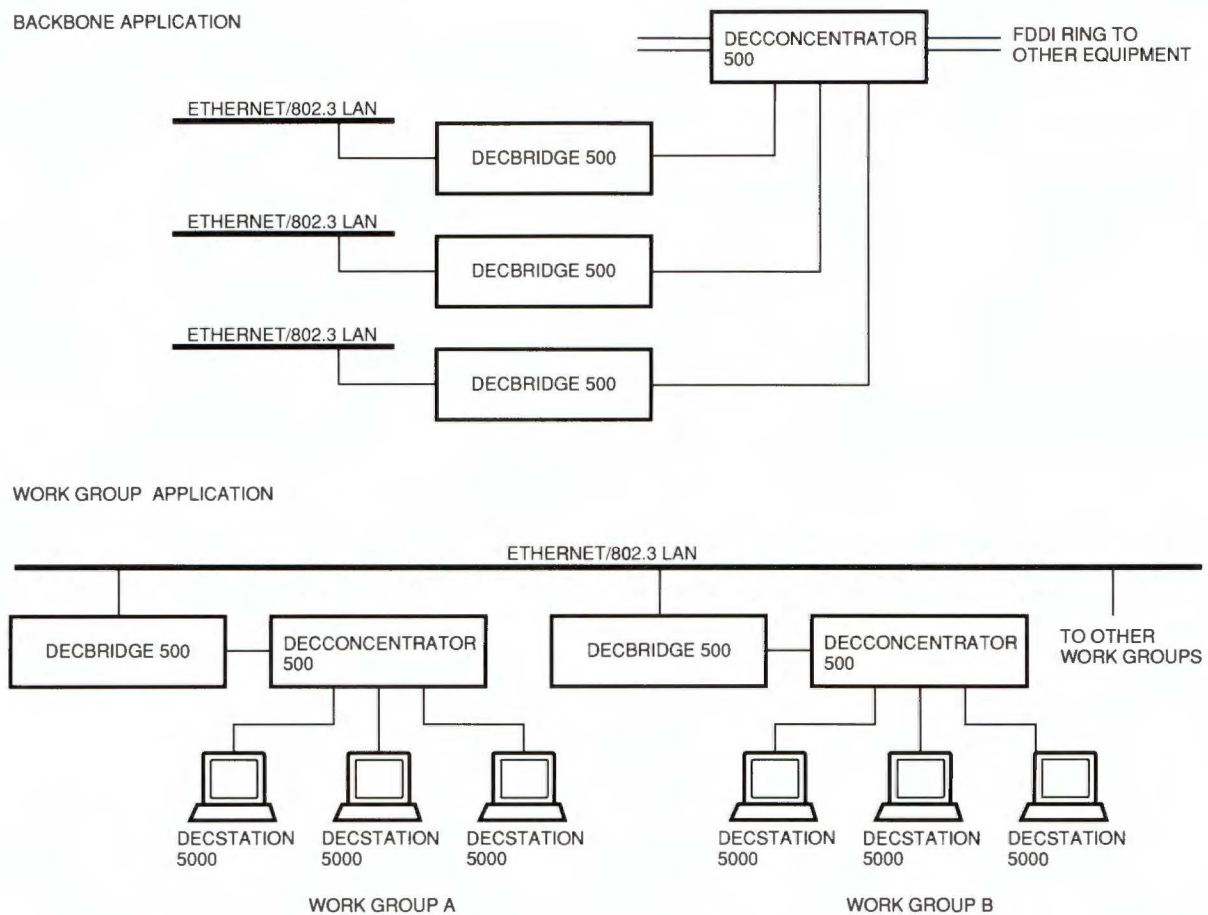


Figure 1 Extended LAN Applications of the DECbridge 500 Product

- Be a proper participant as an end station on both LANs to which it is connected
- Interact with other bridges in the topology of the extended LAN to prevent redundant paths or loops²

Hardware Description

Figure 2 shows a block diagram of the DECbridge 500 hardware design. The applications processor (AP), a subsystem based on a 68020 microprocessor, performs initialization and maintenance of the bridge hardware as well as some steps involved in processing frames. The AP also acts as the management entity for the bridge.

The operating programs for the AP as well as two other processors, the queue manager and the translation processor, are stored in a nonvolatile electrically erasable programmable read-only memory (EEPROM). At initialization, the AP distributes the programs to random access memory (RAM) in the

other two processors' subsystems. The AP executes much of its own program directly from the non-volatile memory, although some high-performance operations are executed from static RAM.

The DECbridge 500 device may have an entire new operating program downloaded over the network and stored in the nonvolatile memory. This allows rapid updates of functionality without the need to perform a hardware upgrade on-site. Program updates are received via either of the attached LANs and stored in an area of RAM referred to as the "landing pad." The AP then transfers the new program into the nonvolatile memory and initiates a firmware reset.

The FDDI and Ethernet/802.3 chip sets and some analog interface circuitry provide connection to the two LANs. The bridge represents a single attachment station (SAS) on the FDDI ring. On the Ethernet/802.3 side of the bridge, switch-selectable ThinWire and attachment unit interface (AUI) connections are provided.

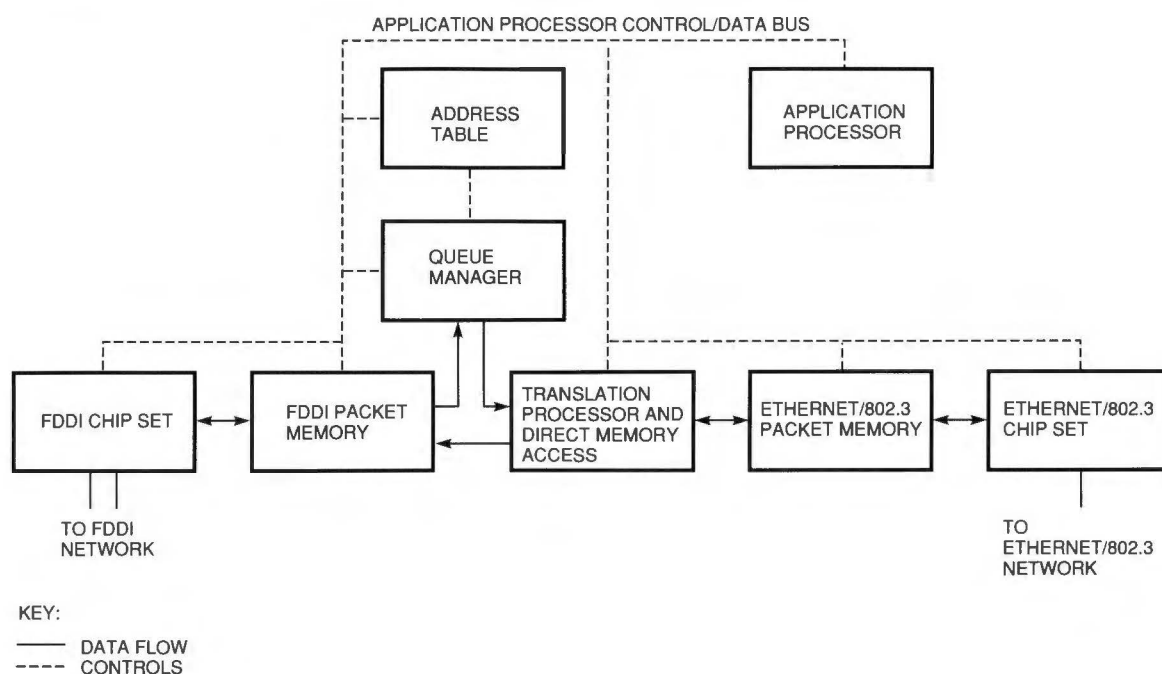


Figure 2 Block Diagram of the DECbridge 500 Hardware Design

Each chip set checks every incoming frame for integrity. Also, some rudimentary identity, or address, tests are applied. Frames that meet the integrity and identity requirements are then placed in a packet memory. The bridge maintains a table of learned MAC addresses. The table contains data for each address that is used to decide if a frame should be forwarded or filtered. The queue manager is a sub-system dedicated to checking each frame received in FDDI packet memory against the information contained in the learned address table. Based on this information, the queue manager decides whether to filter the frame, forward the frame to the Ethernet/802.3, or deliver the frame to the bridge entity for action.

The FDDI and Ethernet/802.3 LANs employ different data link protocols. The translation processor, a second 68020 subsystem, examines frames to be forwarded from one side of the bridge to the other. Each frame is reformatted to the appropriate outbound protocol and moved from the incoming packet memory to the outbound packet memory. The two chip sets examine their respective packet memories for outbound frames and transmit them onto their LANs.

Physical Description

The DECbridge 500 product is shown in Figure 3. The hardware is approximately 7 inches high by 17 inches

wide by 14 inches deep and may be rack-mounted or installed on a tabletop. It operates over the range of 100 to 240 voltage AC (VAC) at 50 or 60 hertz (Hz).

An exploded view of the bridge is shown in Figure 4. The electronics is implemented by the following four logic modules:

- AP, the applications processor
- QM, the queue manager subsystem including the learned address table
- FI, the FDDI chip set and the FDDI packet memory
- NI, the Ethernet/802.3 chip set and packet memory and the translation processor



Figure 3 DECbridge 500 Product

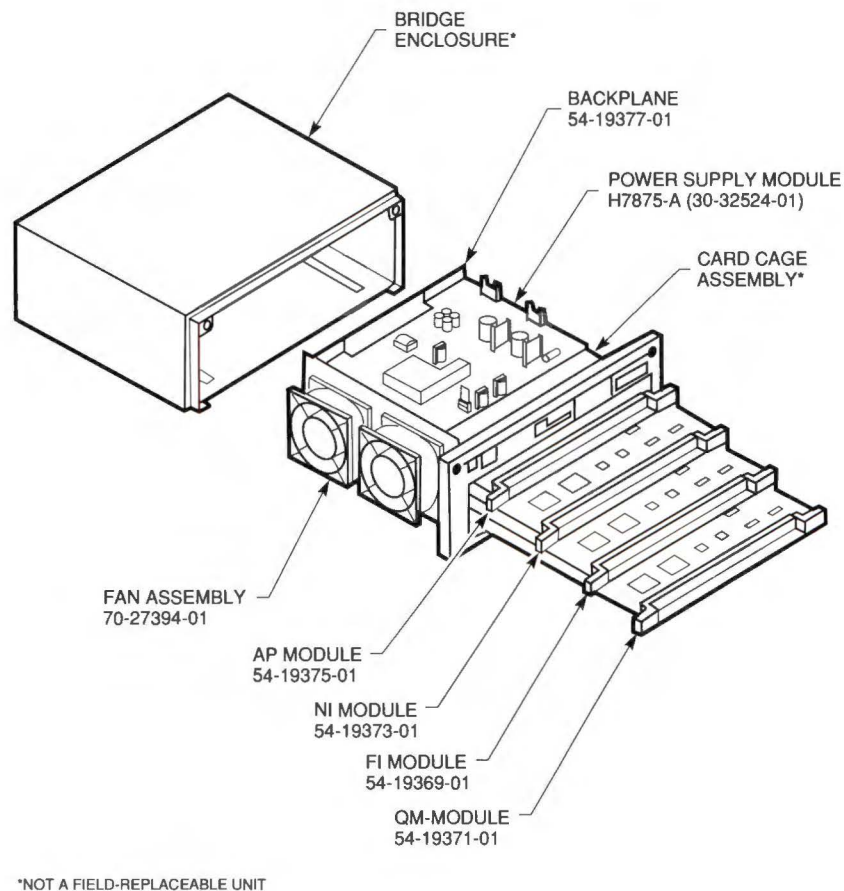


Figure 4 DECbridge 500 Assembly

External signal connectors are located on the front edge of the two network interface cards, FI and NI. Each module has light-emitting diodes (LEDs) for various status functions and also diagnostics. In addition, the AP has a bank of switches for setting certain bridge operating functions.

The power and packaging were designed to simplify the swap out of field-replaceable units (FRUs). The four logic module FRUs can be replaced through the front of the box, without opening it. By taking out only two screws, the outer shell of the case can be removed. This gives access to the three other FRUs, namely, the power supply, the passive backplane, and a fan assembly. The five-sided design of the outer shell results in a product that is mechanically strong and provides shielding from electromagnetic and radio frequency interference (EMI/RFI).

Operation

As mentioned previously, the DECbridge 500 device forwards traffic between two different LAN types. Consequently, the product development team faced

several challenges beyond those encountered in previous bridges that connect similar Ethernet/802.3 LANs. The principal sources of these new challenges were:

- Higher data rates on the FDDI LAN. Ethernet/802.3 operates at 10Mb per second and has a minimum MAC frame size of 64 bytes. The maximum frame arrival rate is 14,880 frames per second (fps). FDDI operates at a rate of 100Mb per second and has a minimum MAC frame size of 17 bytes. The maximum frame arrival rate is 446,429 fps, a rate 30 times greater than that of Ethernet/802.3.
- Different frame formats. Ethernet, 802.3, and FDDI have different MAC frame formats. Traffic entering an FDDI LAN from an Ethernet/802.3 LAN must be properly translated to an FDDI frame format. This translation must be performed in such a way that passage through a second bridge back to a different Ethernet/802.3 LAN results in a frame that recovers its original frame format.

- Different frame sizes. Ethernet and 802.3 differ from FDDI in both maximum and minimum frame sizes. FDDI frames shorter than the Ethernet and 802.3 minimum must be padded. FDDI frames longer than the Ethernet and 802.3 maximum cannot be forwarded, with the exception of special protocol types, which must be broken into multiple, smaller frames.

Objectives

The bridge can only forward frames from the FDDI LAN to the Ethernet/802.3 LAN at the maximum rate accepted by that LAN, i.e., 14,880 fps. Yet the arrival rate of frames from the FDDI LAN may be in excess of 440,000 fps. The incoming frames consist of an unknown mixture of frames that need to be forwarded, frames directed to the bridge itself, and frames to be discarded.

To comply with Digital's bridge architecture specification and the IEEE standard 802.1d for bridges, the bridge must examine all incoming frames.² It must identify, set aside, and process frames of each protocol type directed to itself, in the order received. To meet product performance requirements, the bridge must be able to forward frames at the full Ethernet/802.3 rate. A best effort must be made to buffer frames received in bursts exceeding that rate. Frames should not be erroneously discarded. Results of this compliance visible to the network user are:

- Transparency. Nodes across the extended LAN operate as if they were connected to the same LAN.
- Stability. The paths in the LAN remain constant yet can reconfigure around equipment changes with a minimum loss of connectivity. Frames are not duplicated; nor are they received out of order.
- Manageability. Network management can always observe and control the components of the extended LAN.

The operation of the DECbridge 500 device is best described by examining the progress made through the bridge by frames received from the FDDI LAN. Tracing this flow of traffic also gives insight into many of the challenges faced by the product's development team. The subsystems that process these frames and the flow of frames through logical queues in these subsystems are shown in Figure 5. The operation of the subsystems as the frames progress through them is described sequentially in the following sections.

Receiving FDDI Frames

The FDDI chip set in the bridge places all received frames in the FDDI packet memory on the receive queue. Frames in the FDDI packet memory can be accessed by subsystems in the bridge by using a virtual address method. A page table memory is used to assign a physical 512-byte buffer to each of 16K virtual buffers. Queues contain sequential sets of virtual buffers. Data frames are "moved" from one queue to another by moving the virtual address buffer pointers from one queue to another.

Frames received from the FDDI LAN may be as long as 4500 bytes. Frames longer than 512 bytes are chained, that is, stored in multiple buffers. Each buffer has an associated descriptor longword containing status information about the frame such as error conditions, frame length, and flags indicating the start and end of multibuffer frames. The ability of the bridge to chain small buffers to handle frames of various sizes increases the efficiency of the packet memory by minimizing the amount of unused buffer space. (Statistically, LAN traffic has a higher content of shorter frames.) Thus, more buffers are made available to handle bursts of traffic. Additional information about buffer status is contained in the page table memory. This information is generated and used by the queue manager and the translation processor subsystems.

Queue Manager Process

The queue manager subsystem operates on all frames in the receive queue to determine if they should be discarded, forwarded to the Ethernet/802.3 LAN, or received and processed by the bridge management entity. Discarded frames are returned to the receive queue; the remaining frames are placed on the forward or bridge queues. The queue manager constantly makes updates to the table of learned addresses based on source addresses observed on the FDDI LAN.

The queue manager's operational decisions are based on the following data:

- The frame descriptor containing assorted status information such as transmission errors and frame length
- The frame control field specifying the type of frame
- The type and quantity of frames previously received (used to prevent a flood of any one type of frame from blocking out other types)

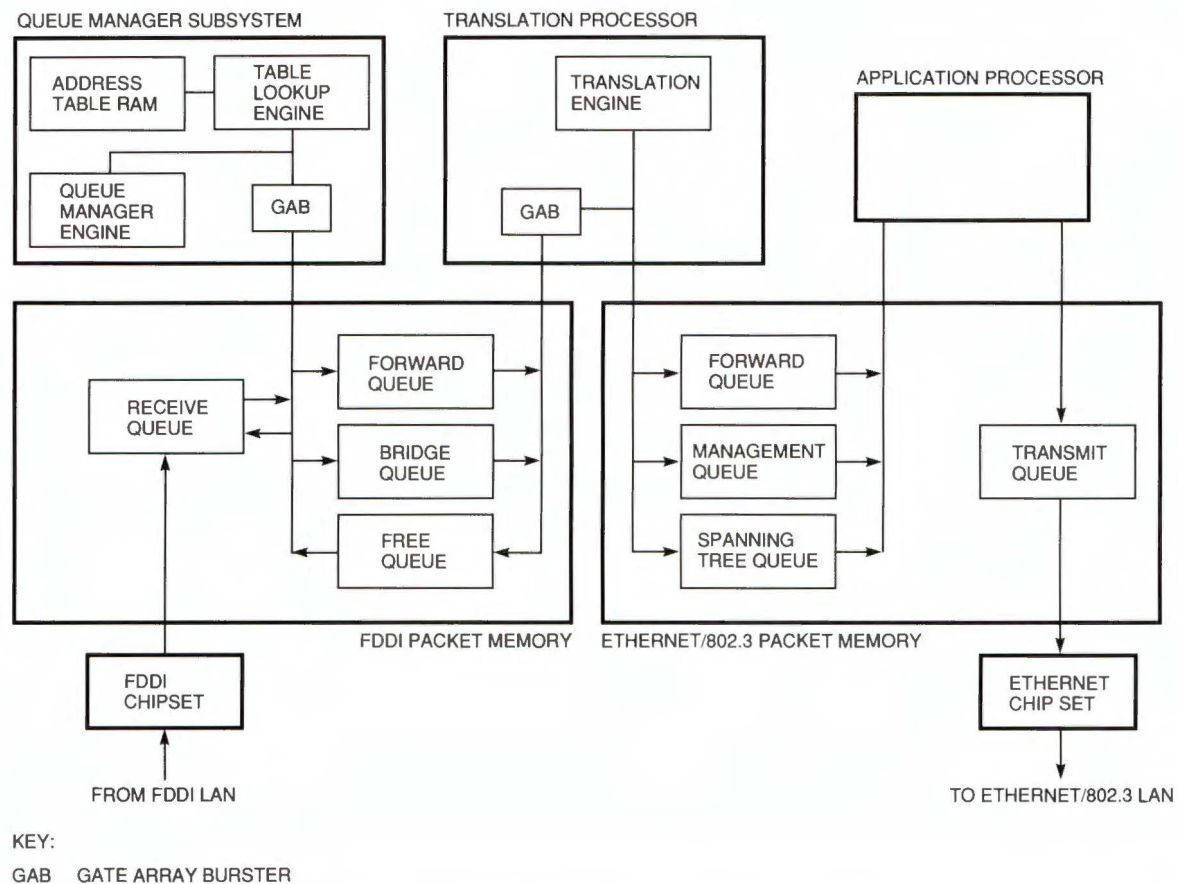


Figure 5 DECbridge 500 Subsystems Flow Diagram

- A learned database containing addresses indicating on which side of the bridge each MAC is located and special filtering status information assigned to each address by network management

The gate array burster (GAB) allows the queue manager to access the FDDI packet memory. This application-specific integrated circuit (ASIC) is a specialized direct memory access (DMA) device. It is capable of moving selected fields or large sections of frames into or out of FDDI packet memory. The objects may be moved either into internal holding registers for examination by the queue manager engine or directly to destinations such as registers in the table lookup engine (TLU). Note that the GAB used in the queue manager subsystem is the same device used in the translation processor, which is discussed later in this paper. These two subsystems have many similar requirements, but each also has unique requirements. Using one GAB design for both subsystems reduced the overall development effort.

The table address RAM and the TLU are key components of the queue manager. The RAM contains a table of up to 16K 48-bit addresses. Each address also has status bits that determine what action the bridge should take when a frame's source or destination address matches a particular address. The TLU is an ASIC with a port that is a slave to the queue manager engine. The queue manager engine inputs an address to the TLU which scans the RAM for that address. If the address is found, the TLU presents that status to the queue manager processor. Otherwise, the TLU gives the queue manager processor a programmable status indicating whether to forward or to discard the frame. A second TLU port allows the TLU and the table address RAM to serve as slaves to the AP. Thus, destination address filtering for traffic received from the Ethernet/802.3 LAN and table maintenance can be performed.

To keep up with the packet arrival rate, the queue manager subsystem makes extensive use of pipelining. The queue manager engine operates

concurrently on six packets. The TLU unit performs three searches concurrently: one each for the source and destination addresses on FDDI packets and one source or destination search on Ethernet/802.3 packets.

Discarding and Keeping Frames The decision to discard a frame is based principally on the frame's address or its contents. The following are typical of frames that are discarded:

- Frames destined for nodes that the bridge recognizes as not on the Ethernet/802.3 side of the LAN. Also, network management may specify addresses to be discarded regardless of location in the topology.
- Frames of either a reserved or undefined frame control type.
- Frames that are either too long or too short.

When a frame is discarded, its buffers are returned to the end of the receive queue by reassigning them in the page table.

Frames that are kept are placed on either the forward or bridge queues. Frames ultimately destined for the Ethernet/802.3 LAN are placed on the forward queue. Frames placed on the bridge queue, to be processed internally by the bridge, are of the following types:

- FDDI station management (SMT) frames
- Digital's extended LAN management software (DECelms) frames or maintenance operation protocol (MOP) frames
- Spanning tree frames, containing messages used to determine the network topology and turn individual bridge ports on or off to eliminate path redundancy
- Frames containing errors

Frames placed on the bridge queue not forwarded to the Ethernet/802.3 LAN. However, after receiving and processing these frames, the bridge may generate one or more frames on either or both LANs. For example, received SMT frames are never forwarded, but a given SMT frame may cause the bridge to transmit additional SMT frames on the FDDI LAN.

Counters Each frame type is guaranteed a minimum amount of processing time by the bridge. If at any time the bridge holds too many of any one frame type, it discards further frames of that type. The queue manager uses allocation counters to keep

track of the number of forwarded, FDDI SMT, bridge management, spanning tree, and error frames.

The queue manager also has counters that summarize its activity. These counters are periodically dumped to the AP and are used to calculate LAN utilization statistics required by network management.

Translation

Bridges operate at and below the data link level in the seven-layer International Standards Organization (ISO)/Open System Interconnection (OSI) reference model shown in Figure 6. The data link layer is divided into a lower MAC sublayer and an upper logical link control (LLC) sublayer. The LLC protocol is specified in ANSI/IEEE standard 802.2.³

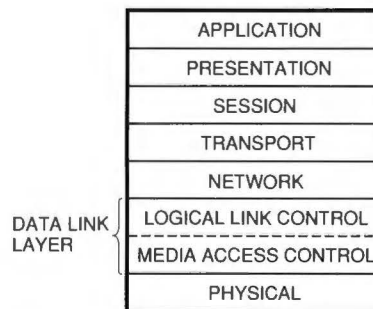


Figure 6 ISO/OSI Seven-layer Reference Model

When forwarding frames from one LAN to another, the DECbridge 500 device converts the outgoing frame to the MAC frame format of that LAN. This process is called translation. Also, when a frame is generated by the DECbridge 500 product on either LAN, the data link frame format of that LAN is employed.

By performing translation, the DECbridge 500 product complies with the IEEE 802.1d requirements for transparent bridging. This enables end nodes to communicate across the extended LAN as if the nodes are directly connected to the same LAN. An alternative to translation, called encapsulation, is possible, but it does not comply with the IEEE 802.1d requirements. Further, using encapsulation puts restrictions on the configuration of the network.

The Process Ethernet, 802.3, and FDDI have different MAC frame formats. When Ethernet or 802.3 frames are bridged to an FDDI LAN, they are reformatted to the FDDI MAC frame format. The original

MAC type (Ethernet or 802.3) is indicated by setting information in the LLC header. If the frame passes through a second FDDI-to-Ethernet/802.3 translation at another bridge, the LLC information is used to determine if the bridge should translate the frame into Ethernet or 802.3 MAC protocol. IEEE standard 802.1 defines a mechanism for translating Ethernet frames into an IEEE 802.2 format (as is used on FDDI LANs). Figure 7 illustrates how Ethernet frames and two types of 802.3 LLC frames are translated into three different types of FDDI data link frames.

Maximum and minimum frame sizes of the LANs also impose requirements on the translation process. Ethernet and 802.3 MAC protocols require a minimum data field length of 46 bytes. The FDDI MAC protocol supports zero-length data fields. When a bridge forwards frames that originated at nodes on an FDDI LAN to an Ethernet/802.3 LAN, the translation process must add padding (null bytes) to any short data fields to bring them up to the 46-byte minimum size.

FDDI has a maximum frame size of 4500 bytes. The Ethernet/802.3 maximum frame size is 1518

bytes. Frames received from the FDDI ring that are longer than 1518 bytes after translation are discarded with the exception of frames discussed in the text that follows.

The internet protocol (IP) is a widely used, network-layer protocol. Nodes on FDDI rings may generate IP frames longer than the Ethernet/802.3 maximum size. The DECbridge 500 product performs one function beyond the process of transparent bridging. The bridge breaks up large IP packets into smaller ones. This function is supported by IP and is called fragmentation. Without fragmentation, the queue manager would discard these long IP frames, preventing communication between nodes on separate FDDI rings that are linked by Ethernet/802.3 LANs.

Since the translation process alters frames, the original cyclic redundancy check (CRC) field is no longer valid. In the DECbridge 500 device, the translation process concurrently verifies the received CRC, translates the frame, and generates a new CRC. This concurrent processing results in a high degree of data integrity.

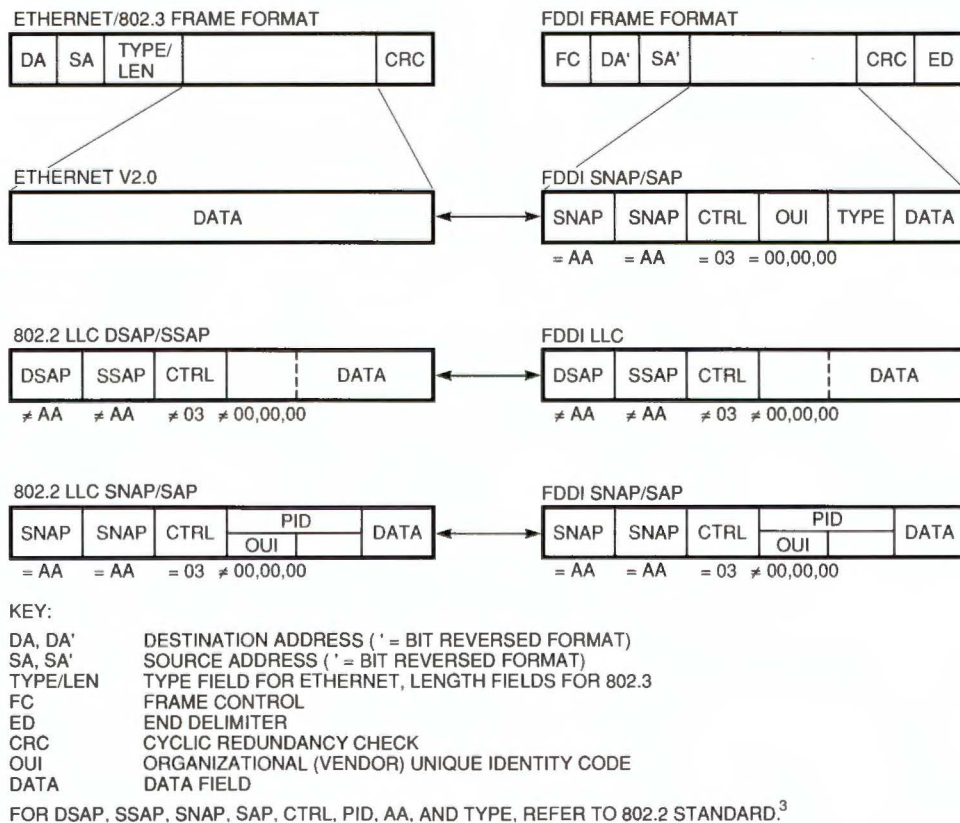


Figure 7 Translation of Ethernet/802.3 Frames to FDDI Frames

Address Bit-ordering The bits of the destination and source addresses are transmitted in the reverse order on FDDI from that on Ethernet/802.3 data links. Digital's FDDI chip set performs a bit-reversal operation on receive and transmit for only the MAC frames' destination and source address fields. Since these MAC fields are stored inside the bridge in IEEE 802.1, canonical bit-ordering, only one version of each address needs to be kept in the forwarding database. Also, when generating management messages, this method of frame storage allows the bridge to move an address from the source or destination field of a received frame into the data field of the management message without modification. When calculating the CRC on incoming packets, or generating a new CRC on forwarded packets, the translation process must take into account the bit-ordering.

Implementation The translation processor consists of principally a GAB and a translation engine (based on a 68020 subsystem). The GAB and translation engine interactively copy frames from FDDI frame memory to Ethernet/802.3 frame memory. Concurrently, the translation engine makes changes to the frame format, and the GAB calculates both the CRC of the incoming frame, using old bit-ordering, and generates the CRC of the translated frame, using both new bit-ordering and new frame format.

Frames from the forward queue in FDDI frame memory are, thus, translated and moved to the forward queue in Ethernet frame memory. Frames from the bridge queue are separated into management and spanning tree queues in Ethernet/802.3 frame memory. The translation processor returns buffers from the forward and bridge queues to the free queue in FDDI frame memory. The queue manager returns buffers from the free queue to the receive queue, making them available to store newly received frames.

NI-side Processing

Frames placed in output queues by the translation processor are processed next by the AP. Frames in the spanning tree and management queues are destined for the bridge as a manageable entity on the extended LAN. The AP processes these frames and may generate response traffic on either the FDDI or the Ethernet/802.3 LAN.

Frames in the forward queue are subjected to additional match or nonmatch filtering by the AP. These frames are checked against a list of protocol types loaded by network management (e.g., TCP/IP and AppleTalk protocols). Protocol filtering is often

a useful mechanism to prevent all frames of one or more protocol types from propagating across the extended LAN. The AP also uses the table lookup engine to check frames against a list of source addresses loaded in the address table RAM for filter/forward requirements. Source address filtering may be used to contain traffic from nodes with an unusually high transmit rate. In addition, this filtering may be used as a form of security to deny access to nodes that are masquerading, that is, transmitting by using another node's address.

The bridge checks frames against protocol and source address lists after the frames have been filtered by destination address in the queue manager. The rate of frames here is lower, not exceeding the Ethernet/802.3 LAN rate. Performing such checking on all incoming traffic from the FDDI LAN would require significant additional computational work by the queue manager subsystem.

The frames in the forward queue that pass the protocol and source address filters are placed on the transmit queue of the Ethernet/802.3 chip set. The AP must merge these frames into the transmit queue with management traffic that the AP has generated for the Ethernet/802.3 LAN.

NI-to-FDDI Forwarding

The bridge processes traffic received from the Ethernet/802.3 LAN in much the same way as FDDI LAN traffic processing was described in the preceding material. It is essentially a mirror-image process, but a few significant differences exist.

The lower arrival rate of frames from the Ethernet/802.3 LAN does not require a dedicated frame-processing engine such as the queue manager. Thus, destination address filtering is performed by the AP, which shares the TLU engine and table address RAM with the queue manager.

Also, allocation counters are not used on Ethernet/802.3 traffic. The AP directs all incoming traffic into different queues at full rate. Unusually high bursts of a particular frame type could overflow a given queue.

Another difference is a requirement for stations placing traffic on the FDDI ring. On token ring networks, the transmitting station is responsible for removing its own frames from the ring. A typical station knows which frames to strip by recognizing its own address as the source address. When a bridge transmits a frame, the source address is that of the originating node. In the DECbridge 500 product, the stripping function is handled in the FDDI chip set. A bridge strip algorithm is implemented

that generates frames marking the end of a block of transmitted frames and also makes use of counters for sent and stripped frames.

Development Methodology

It was important to get the DECbridge 500 product to market in as short a time as possible. The solidification of the ANSI FDDI specifications, coupled with the appearance of products from different vendors, created a finite window of opportunity. At the same time, the requirements to be met were significant. The next three sections present brief descriptions of some elements of the development methodology that were employed to meet the design requirements while optimizing the development schedule.

Utilization of Existing Technology

The DECbridge 500 developers combined technology from existing products with their own new technology in the following design areas: system level, electrical, firmware, and mechanical and power.

System-level Design The AP and the Ethernet/802.3 packet memory of the DECbridge 500 product correspond approximately to the processor and memory of its most recent predecessor, the LAN Bridge 200. The high FDDI packet rate required the use of a separate processor to filter incoming FDDI traffic. Also, another dedicated processor was necessary to perform the translation function. (Ethernet-to-Ethernet bridges do not require a translation function.) The resulting increase in the rate at which a processor accesses frame data required the development of a separate packet memory for the FDDI LAN.

Electrical Design The Ethernet interface and packet memory designs again were borrowed from the LAN Bridge 200 product, but several extensions were needed. The AP design is very similar to the design of the processor in the LAN Bridge 200, but it has several new features, namely, a down-line, loadable program memory, a bus system for communicating over the backplane with other modules, and a distributed interrupt system. The queue manager, the translation processor, and the FDDI chip set with packet memory are new designs. The additional circuitry resulted in a multimodule system with a backplane.

Firmware Design The DECbridge 500 product uses the same operating system as other Telecommunications and Network products. Much of the

firmware associated with the bridge entity and with Ethernet-side processing was modified from the LAN Bridge 200 product. The queue manager and translation processor required all new code.

Mechanical and Power Designs Previous products typically consisted of a single module mounted in a box. The DECbridge 500 device required developing a multimodule system with a backplane. The initial goals were to install two, or at most three, logic modules. To minimize the risk to the module development schedule, a four-board approach was adopted, which closely follows the block diagram shown in Figure 2. The box, the backplane, and the power supply are all new designs.

Integration of FDDI Products and Chip Set Development

A strategy was adopted to maximize the commonality of effort in the development of the DECbridge 500 and the DECconcentrator 500 products, and in the evaluation of the FDDI chip set. When a product set was defined, plans were in place to develop a hardware test bed for the FDDI chip set. The test bed design was expanded midstream so that separate modules could be added, turning it into a breadboard for either a DECconcentrator or a DECbridge device. The two DECbridge modules contained the queue manager, the translation processor, and the Ethernet/802.3 chip set and packet memory. The test bed provided the FDDI interface, an FDDI packet memory, and an application processor, as well as a power/packaging platform. While evaluation of the breadboards was still taking place, activities were accelerated to develop the products.

Technical Risk Analysis

Different approaches were adopted for various parts of the bridge design based on technical risk. Completely new technology, e.g., the queue manager and the translation processor, were simulated, breadboarded, and tested. Areas that were understood but still new, e.g., packet memory designs, were evaluated largely by gate-level simulation. High-confidence areas, such as designs taken from previous products, were evaluated in the prototype products.

The DECbridge 500 product employs three processors. Thus, a lot of the bridge functionality was in firmware, and changes could be made with relatively little impact on the schedule. Also, in several instances, deficiencies found in the system-level design could be corrected in the firmware.

Use of Parallel Activities

Several parts of the usual development process were overlapped to minimize time. A combined functional and design specification was generated instead of going through two serial stages to produce separate specifications. In the hardware design, module layout started once a confidence factor was achieved through simulation. Design reviews were held concurrently with module layout, and performance simulation continued throughout the process. There was a close interaction of the printed circuit board layout group and the electrical designers.

In product qualification, a pipelined system of reliability qualification testing (RQT), process qualification testing (PQT), and internal/external field test was set up to accommodate a phased-release of firmware. RQT and PQT started with a functional, subset release of the firmware. Hardware confidence grew. After electrical design verification testing, firmware with the minimal functionality for field test was tested briefly in RQT and PQT and then shipped to the field. New firmware releases were developed with increased functionality as well as corrections to recognized problems. A process was developed whereby new releases were tested for a few days each in RQT and at internal field test sites and then released to external field test sites. The down-line-upgrade ability was instrumental in allowing us to use this process.

Conclusions

Differences in frame format, frame length, and transmission speed place requirements on an Ethernet/802.3-to-FDDI bridge that are not encountered in bridges between like data links. The DECbridge 500 product met these requirements by dedicating one processor subsystem to the translation process and another to the process of filtering and sorting incoming FDDI frames. By adhering to the requirements of the IEEE standard for transparent bridging, the DECbridge 500 device allows the problem-free interconnection of FDDI LANs to the large existing base of Ethernet/802.3 LANs.

The development team concluded that by performing risk analysis and having backup plans in place, several parts of the standard design process could be compressed or overlapped. Fundamental to the design was the ability to make remote, nonvolatile upgrades to the product's operating firmware.

Acknowledgments

A large number of people contributed to the success of the DECbridge 500 project. It is not possible to name all of these important contributors in the space provided. Therefore, the authors wish to acknowledge that the DECbridge 500 product could not have been developed without significant contributions from members of the Telecommunications and Networks (T&N) Communications Systems Engineering Group responsible for the diagnostics, firmware, hardware, product assurance, and software and from the following T&N organizations: Architecture, Customer Services Systems Engineering, Marketing, Manufacturing, Product Management, and Publications.

References

1. F. Backes, "Transparent Bridges for Interconnection of IEEE LANs," *IEEE Network*, vol. 2, no. 1 (January 1988).
2. *Local Area Network MAC (Media Access Control) Bridges*, IEEE Standard 802.1(d) (New York: The Institute of Electrical and Electronic Engineers, Inc., 1990).
3. *Logical Link Control*, ANSI/IEEE Standard 802.2-1985, ISO/DIS 8802/2 (New York: The Institute of Electrical and Electronics Engineers, Inc., 1985).

The DECconcentrator 500 Product

Digital's decision to implement the fiber distributed data interface (FDDI) physical topology with a dual ring of trees, as opposed to a dual ring only, resulted in the development of the DECconcentrator 500 product. The dual ring of trees topology provides high availability, manageability, and support for building wiring standards. The function of the concentrator demanded that the product be reliable, provide for remote management and control, and allow a low cost per connection. The use of common FDDI hardware and software components developed by Digital helped the product team to meet these goals.

Concentrators in the ANSI FDDI X3T9.5 Standard

In the initial stages of its development, the American National Standards Institute (ANSI) standard for the fiber distributed data interface (FDDI) technology was intended for a computer room system interconnect, similar to Digital's Computer Interconnect (CI) bus. All stations were to be dual attachment stations (DASS) and the interconnections between the stations were to be wired directly, without patch panels or similar structured wiring schemes.

Using this dual ring topology is feasible for a small number of stations in a single, tightly controlled room. However, soon the emphasis of FDDI shifted primarily to local area networks (LANs). A LAN consists of many nodes, spread over a large area and with potentially many individuals able to connect and disconnect stations.

To accommodate LAN topology requirements, ANSI chose to add the concept of concentrators midway in the development of the FDDI standard.¹ In the simplest case, a concentrator is a device that attaches to a dual ring (via A and B ports) and provides additional ports (M ports) to which stations can be connected by means of radially wired cables. These additional stations can be single attachment stations (SASS) with a single port (S port) rather than the pair of ports required by a DAS. This simple topology was soon generalized, allowing concentrators to be nested to any depth in a dual ring of trees. Concentrators may be singly attached (by using an S port plus M ports rather than A and B ports plus M ports), and DASS may be connected to concentrators. Figure 1 illustrates the basic FDDI topologies.

FDDI concentrators are more than wiring hubs, unlike certain other LAN technologies. They also perform two functions that are key to network integrity. When a station's connection to a concentrator is activated, a multistep initialization procedure called physical connection management (PCM) takes place, using physical layer (PHY) signaling. In this procedure, the station and the concentrator exchange some topology information, and a link confidence test is performed to verify that the data integrity on the link is acceptable. Once the PCM initialization is complete, the connection becomes part of the ring.

Also, concentrators continuously perform link error monitoring (LEM). Each active link is monitored for data errors, and a link found to have excessive data errors is disabled. In this way concentrators ensure that the ring error rate, and therefore the packet loss rate, remains acceptably low.

Given the many choices for concentrator interconnection allowed by the ANSI standard, it is possible to construct highly complex topologies, including many that have "bad" properties. When a station is physically plugged in and that connection is operating properly, the station should be able to communicate with all other stations in its own network. This property is often stated as "Physical connectivity equals logical connectivity," or, in other words, "Being plugged in implies being able to communicate." In bad topologies, this property does not hold. Such topologies are very confusing to network managers and are therefore undesirable.

The ANSI standard specifies some topology rules to reduce this problem. However, the decision to

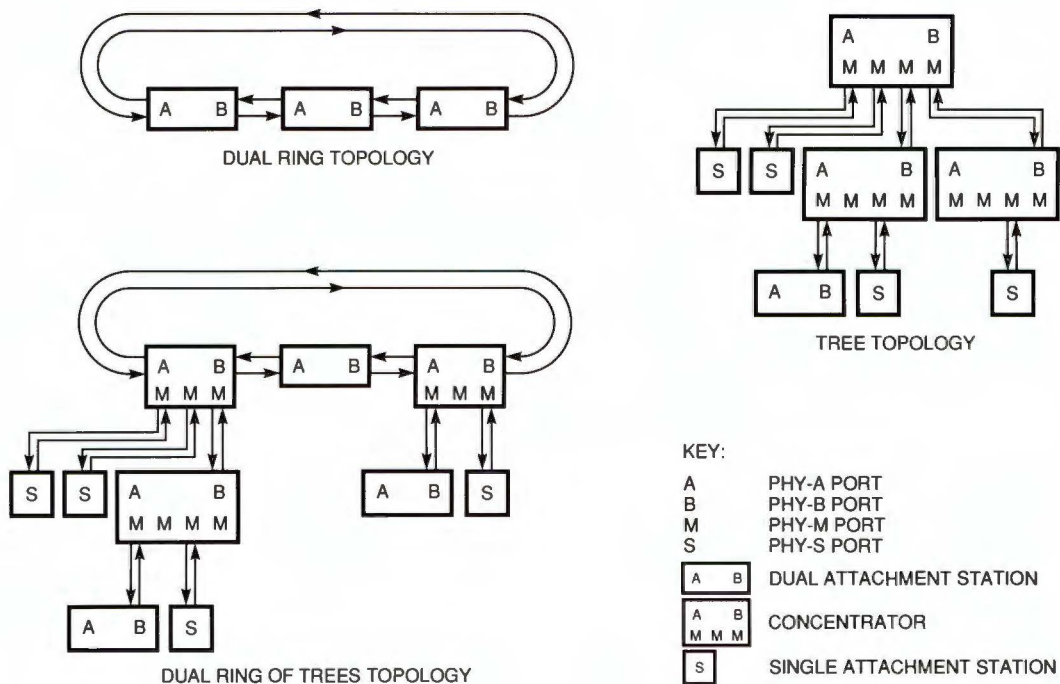


Figure 1 Basic FDDI Topologies

accept or reject offered connections is based only on local knowledge (i.e., information held locally in each station or concentrator), and it is not possible to detect all the bad topologies. FDDI therefore depends to some extent on the competence of the network manager to avoid bad topologies.

One special case allowed by the topology rules is called dual homing, as shown in Figure 2. With a dual homing configuration, the A and B ports of a concentrator or a DAS are connected to M ports, usually of two different concentrators. In this case,

the B to M connection becomes active, and the A to M connection remains in a "standby" state. The standby connection is not part of the ring, but it can quickly change to the active state if the B to M connection breaks. In this way, connectivity is maintained when failures occur.

We next present the reasons Digital chose the dual ring of trees topology over the dual ring topology and examine the resultant need for a concentrator. A detailed discussion of the development of the DECconcentrator 500 product follows.

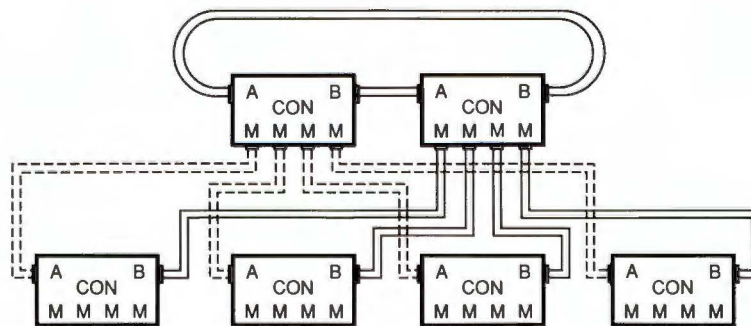


Figure 2 Dual Homing Tree

Digital's Choice of the Dual Ring of Trees Topology

A dual ring topology consisting of dual attachment stations may be appropriate in very small networks that do not use structured and permanent cable plants. But a dual ring of trees topology, using single attachment stations as the end-user devices and concentrators as hubs, provides the optimal solution for a highly flexible, reliable, available, maintainable, and robust FDDI LAN. As described in the preceding section, the concentrator is the cornerstone of an FDDI network. Its significance in building large, robust, and most importantly manageable FDDI networks has been recognized by many of our key customers. Digital chose the tree/dual ring of trees architecture implemented with concentrators over a purely dual ring approach because this architecture offers:

- Support for industry standard radial wiring practices
- Manageability
- Configuration flexibility
- A definable demarcation point between the end user and the backbone
- Scalability

Although the specific behavior of an FDDI concentrator is relatively new, the concept is not. Most system vendors and users of large systems have adopted the use of manageable hubs (multiport repeaters) for Ethernet networks and media access units (a type of passive concentrator composed of bypass relays) in token ring networks.

Initially, cost was a major concern in the decision to implement a tree-type architecture. Some users saw the addition of the concentrator as an added cost burden. However, the cost increase, which can be amortized over the entire network, is greatly outweighed by the added advantages. With a concentrator, stations can be separated into two categories, end-user devices and backbone devices. This is especially important in large networks where the functions of network administration and users of the LAN are totally disparate. The concentrator becomes a tool of the network managers that simplifies their role. As a demarcation point between end-user devices and the backbone, the concentrator protects the backbone from inadvertent disruption caused by the end user.

As shown in Figure 3, the actual number of components that are required to connect eight FDDI stations, whether into the dual ring or in the tree, is approximately the same; only the distribution of these components is different.

In both topologies we have eight physical connections. A physical connection, whether in the tree or in the dual ring, is a point-to-point, full-duplex path between adjacent physical layers. The initial reasoning for a dual, counter rotating ring was to create a bidirectional data path between adjacent stations in which the secondary path's main purpose is to assist in startup, initialization, and reconfiguration of the primary ring.

Either topology requires roughly the same number of components, i.e., PHY entities, optical transceivers, cables, etc. With the tree, the PHYs are rearranged so that conceptually we have taken a

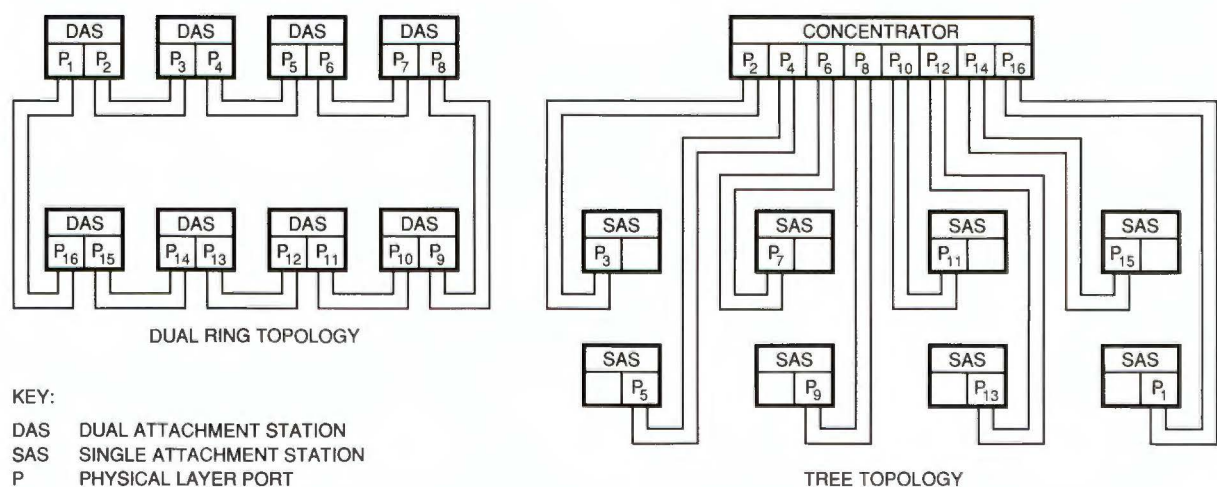


Figure 3 Components of Two FDDI Physical Topologies

PHY entity from each DAS, in conjunction with a data path switch, to create the concentrator. This approach does incur the additional cost of the power and packaging. However, from the perspective of the network administrator, this small incremental cost is offset by the increased ability to manage and control the network, made possible by the concentrator.

A large network of many stations has a high probability of disruptions. Although a DAS and a concentrator are fundamentally different, they have in common the role of controlling the network topology through PHY-level signaling. In the case of a disruption, whether an operator-initiated function (i.e., a station powered down, installed, or removed) or a failure of the station or cable, the token path is modified according to the station management/connection management (SMT/CMT) algorithm to maintain a continuous logical ring.

The main difference between the two approaches to solving the disruption problem is in the way a station is bypassed. With the dual ring approach shown in Figure 4, when a disruption occurs, the stations adjacent to the disruption bypass the offending station(s) and reconfigure the ring by wrapping the secondary and primary rings to form a new single continuous ring. This provides a degree of fault tolerance but is limited to only a single disruption.

In the case of multiple failures or disruptions, all dual attachment stations adjacent to the faults reconfigure, thereby creating multiple disjoint rings. Even though the majority of the stations in the network might be operational, they would operate over several disjoint networks. The potential loss of the service access point would effectively leave the network nonoperational from the client/server perspective. Management of such a situation would

also be an ordeal, since access to fault information would be limited to the stations remaining on the portion of the ring to which the management station was directly attached.

An FDDI concentrator provides fault tolerance in a different way, as illustrated in Figure 5. When a station connected to a concentrator is removed or powered off, the failure is bypassed through the concentrator data path switch at the PHY level. Any one or all of the stations can be effectively bypassed through the concentrator without affecting the connectivity of the other stations or the global topology of the FDDI network.

Structured Wiring

To fully appreciate the benefits of the concentrator, let us consider an FDDI network implemented in an office building environment in conjunction with structured wiring.

A typical building environment includes wiring between offices and equipment rooms on the same floor and in between floors. The wiring is permanent and involves a relatively large number of end-user devices as well as backbone devices over moderate distances. Moreover, frequent adds/moves/changes occur in this environment, and the ability to move from one location to another without manual intervention or network disruption is desirable. A clear demarcation between end-user devices and backbone is required to maintain the integrity of the backbone and to minimize disruption to, or manipulation of, the backbone cabling.

The Telecommunication Industries Association (TIA), together with the Electronics Industries Association (EIA), is defining a commercial building wiring standard; draft EIA/TIA 568 has the framework as designated in Figure 6.² According to this standard, end-user devices in offices should be

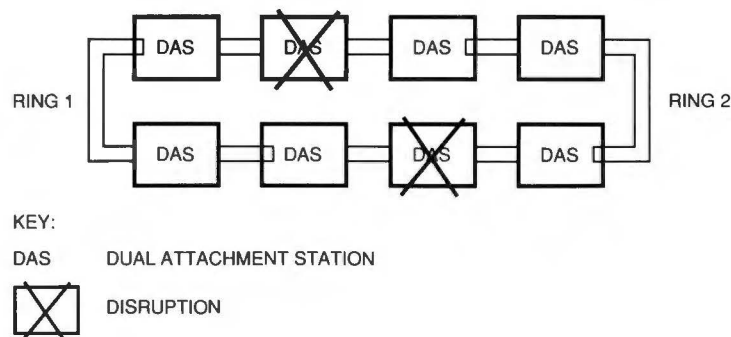


Figure 4 Disruptions in a Dual Ring Topology

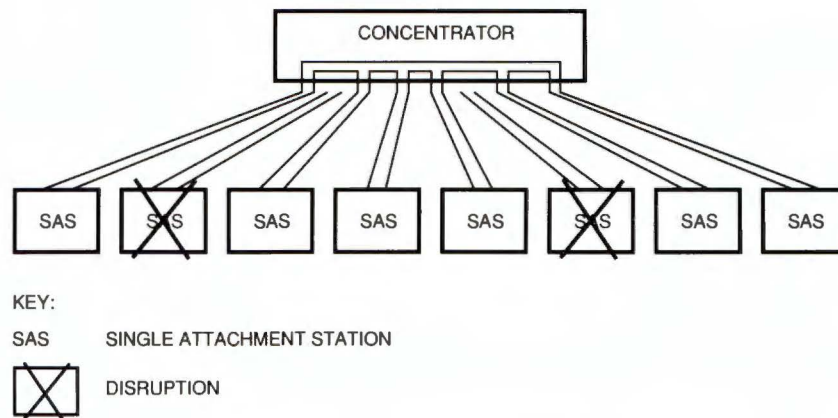


Figure 5 Disruptions in a Tree Topology

wired from a telecommunications closet (TC). All closets in each building then connect to the intermediate cross connect (IC) or to the building hub for that building. In turn, all building hubs connect to a single main cross connect (MC) or to the campus hub in the campus.

Single Attachment Stations

The tree topology, as illustrated in Figure 7, facilitates structured (or radial) wiring as prescribed by the draft EIA/TIA 568 standard. The end-user devices, implemented as SASS or DASS, connect to

the concentrators located in telecommunications closets, which are maintained and controlled by the network administrator. When concentrators are used, the most cost-effective user stations are SASS. Connection to concentrators keeps the end-user devices separate from the backbone so that a disruption in an end-user device, such as disconnecting a station, does not affect the operation of the network. The concentrators in the various closets connect to root concentrators in the building hubs. If other backbone devices are present in the building hubs or communication closets, such as bridges,

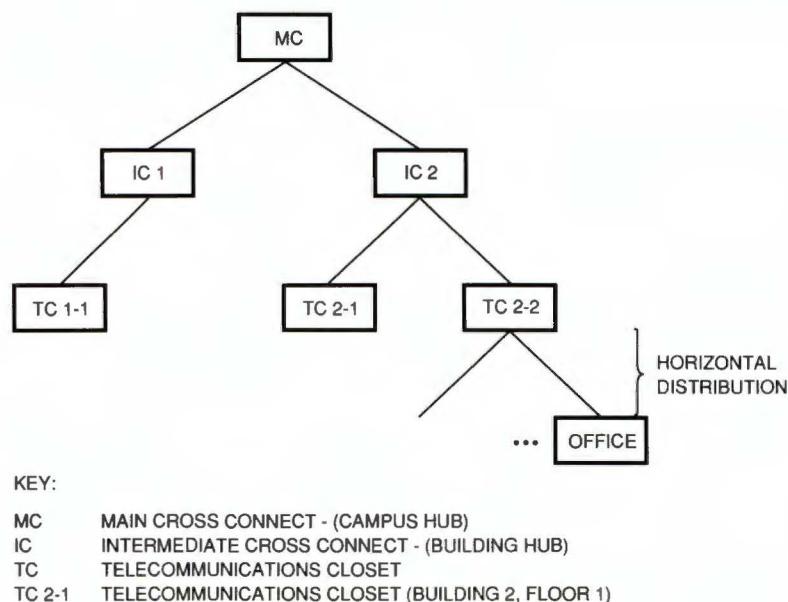


Figure 6 Draft EIA/TIA Building Wiring Standard

they can also be connected to concentrators in the building hubs or the campus hub, as appropriate. The tree topology offers the fault tolerance as well as configuration flexibility required in a structured wiring system. Also, it allows for adds/moves/changes without disrupting or manipulating the backbone cabling.

Dual Attachment Stations

End-user devices directly attached to the dual ring, however, are not easy to isolate from the backbone LAN. Both the end-user devices and the backbone devices are part of the same physical loop, as shown in Figure 8. To a network administrator, management and control of the backbone becomes an ever increasing ordeal because each end-user station is now considered part of the backbone. Even though rules for the end-user behavior can be established, they cannot easily be enforced.

The availability of the backbone is increased by the use of concentrators, since these are the only devices that form the dual ring backbone. This benefit is very important for a large network. For example, in a network supporting 200 end-user stations on a dual ring of trees topology, if 8-port concentrators are used to connect them to the dual ring backbone, only 25 concentrators reside on the dual ring backbone. The reconfiguration of the backbone is dependent on only 25 devices. Also, the network administrator needs to control only 25 devices. In contrast, if the same 200 stations are DASs directly attached to the dual ring, the reconfiguration of the backbone is dependent on 200 devices. The probability of having two or more disjoint rings is much

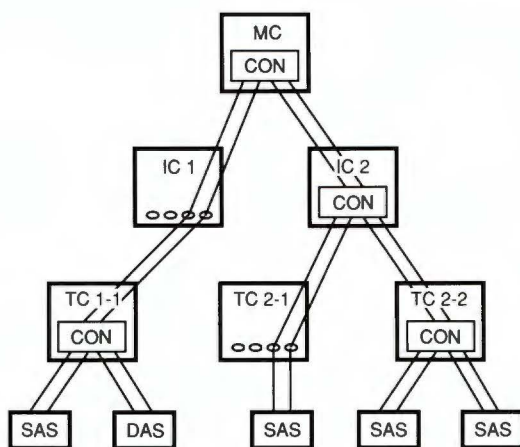


Figure 7 Implementation of a Tree Topology in Structured Wiring

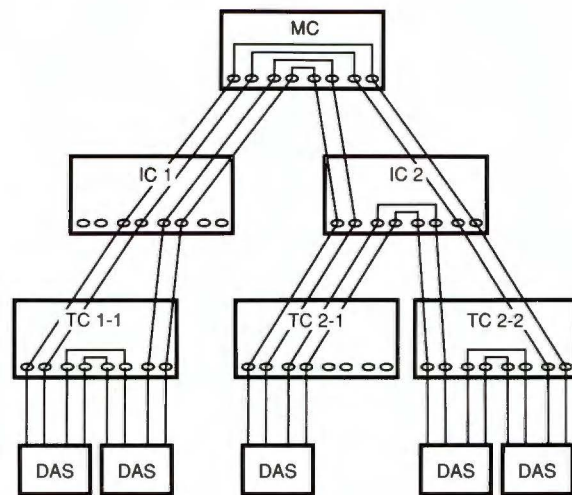


Figure 8 Implementation of a Dual Ring Topology in Structured Wiring

higher in the latter case. Also, with DAS stations, the network administrator is faced with the impractical task of directly controlling 200 devices.

Fault-tolerant Configuration Options

Two fault-tolerant configuration options are available: bypass relays and dual homing. Bypass relays may be used with DASs directly attached to the dual ring to provide fault tolerance in addition to the single fault protection provided by wrapping to the secondary ring. Dual homing is an alternative mechanism which allows dual attachment devices to have a redundant connection to a concentrator when installed in a tree topology. These two alternatives are examined in this section.

Bypass Relays

To avoid the aforementioned reconfiguration problems with DAS implementations, the FDDI standard offers an option of using an optical bypass relay. While such relays are envisioned to alleviate some of the reconfiguration problems, they may induce more problems than they solve. The inclusion of relays in the network means added cost of components, cables, and connectors, loss of optical power, reduction in interstation distance, and an additional failure mechanism. These factors limit the use of such relays to possibly very small, physically collocated work group LANs and make the relay an unattractive solution for a large network environment.

As shown in Figure 9, the end-user stations A, B, and C are dual attachment with bypass relays, and

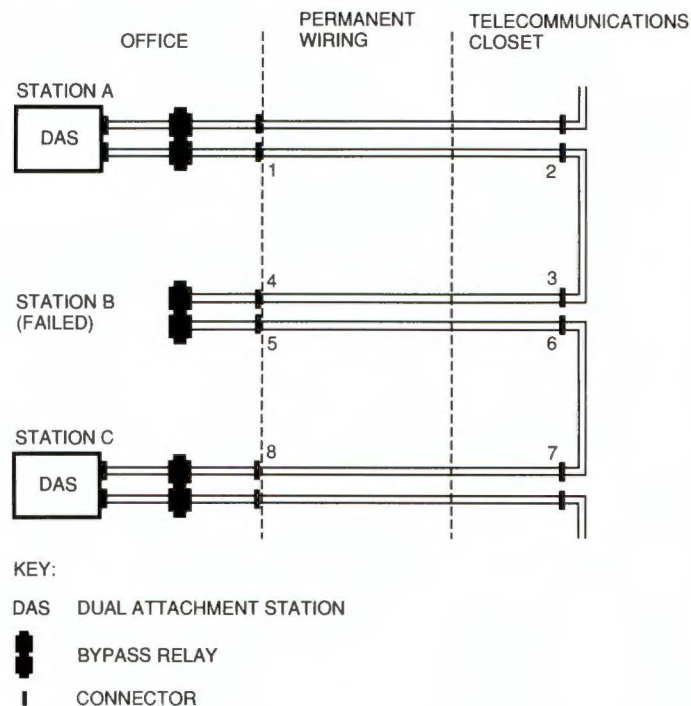


Figure 9 Bypass Relays

station B has failed. With B bypassed, stations A and C become adjacent. The total link loss between these two stations, using a fiber-optic cable of 62.5-micron core diameter and 125-micron cladding diameter, must not exceed 11 decibels (dB) to comply with the FDDI standard.

Let us assume that the fiber-optic cable has a loss of 1.5 dB per kilometer (according to the EIA/TIA-492), and the distance between the communication closet and each station is 50 meters.³ Since the cable length between stations A and C is 200 m, the power loss is 0.3 dB. A connector has a loss of 0.7 dB (according to the EIA). Since there are eight connectors (labeled 1 through 8 in Figure 9) between A and C, the power loss is 5.6 dB. A bypass relay has a loss of 2.5 dB. Since there are three relays between A and C, the power loss is 7.5 dB. The total link loss between stations A and C, therefore, is 13.4 dB, which is in excess of the maximum allowed by the FDDI standard. Note that with the use of optical bypass relays, the effective distance is limited to less than 200 m, which is far below the maximum of 2 km allowed by the standard.

Dual Homing

For some applications a tree or dual ring of trees may not meet the customer's requirements. The dual

homing topology, shown in Figure 2 and described earlier in the paper, has none of the limitations or problems that can be imposed by the dual ring.

This topology is beneficial in a large campus to connect remote buildings into the FDDI backbone. It allows standard radial wiring practices, with up to 2 km between the campus hub and any given building through multimode fiber (MMF) links. The dual homing topology is especially useful with long distance links utilizing single mode fiber (SMF), which span distances of up to 40 km. With the dual ring topology each span has to be counted four times towards the fiber-optic path length maximum of 200 km allowed by the standard, in the event that a wrap occurs. For a link of 40 km, 160 km has to be subtracted from the maximum of 200 km, thus leaving only 40 km for the rest of the network. Using a tree topology, the span is counted twice as it has only one active fiber pair.

Product Development

The following sections examine the DECconcentrator 500 product development process from beginning to end, elaborating on details of the product functionality as they were refined along the way.

Several key factors provided a smooth product development process. First, the architecture for

the DECconcentrator 500 product was chosen as a subset of the generality allowed by the ANSI FDDI standard. Several features which would have significantly complicated the product without greatly enhancing functionality were not included. Examples of these are "roving MAC" and the ability to allow stations to select the ring (primary or secondary) to which to attach. Second, product management established a clear list of priorities, requirements, and goals that allowed the development team members to focus their efforts. The absence of significant changes in architecture or product requirements during the development helped the team stay on schedule. The priorities selected were to design for low cost and high reliability, provide for ease of firmware upgrades, and strive for quick time to market. The emphasis on simplicity and reliability enabled us to keep product transfer cost to a minimum and to nearly meet the time-to-market goal. And finally, the relatively small and tightly knit development team stayed together from conception through field test and first product shipment.

Hardware Partitioning

As the hardware block diagrams were developed, several concepts for partitioning the hardware into modules were evaluated. The electrical, mechanical, and power supply designers worked closely together to choose a suitable partitioning.

The initial high cost of the FDDI fiber-optic transceivers led the designers to select modular hardware partitioning. A modular design allows ports to be added according to the customer's needs, thereby minimizing both the initial cost and the number of unused ports. Since Digital's networking products have traditionally used side-to-side airflow for cooling, a card cage that supported horizontal modules was chosen. Four ports per module was considered to be a reasonable number by which a customer could increment a system. This module, referred to as the port module, contains four sets of the FDDI physical layer chip sets, one status light-emitting diode (LED) per port, one module field-replaceable unit (FRU) fault LED, and a small amount of support logic.

One function of the DECconcentrator 500 product is to provide support for network management; this requires data link layer hardware. In addition, the DECconcentrator 500 device must connect to the FDDI dual ring; this requires the A and B port types. The DECconcentrator 500 management module was designed to meet these needs. By combin-

ing the data link and A and B PHY port hardware, we ensured that any DECconcentrator 500 device installed in a dual ring of trees would be manageable.

The DECconcentrator 500 product also includes a microprocessor to execute diagnostic and operational firmware. To minimize the number of modules, we specified that the microprocessor and its support logic fit on the backplane. The use of an active backplane eliminates the need for a separate control module in the card cage, thereby reducing both cost and the vertical height of the box. The backplane also provides the token ring data path function which interconnects any allowable configuration of port and management modules. The number of modules supported by the backplane is based on our evaluation of customer need. We decided that 8 to 12 ports per concentrator is sufficient for most end-user configurations. Two basic configurations are supported in the DECconcentrator 500 product. A concentrator configured with one to three port modules (4 to 12 ports) can support a standalone work group but cannot connect in the dual ring of trees topology. A concentrator configured with a management module and one or two port modules supports the dual ring of trees topology and is remotely manageable.

Another goal of the hardware team was to eliminate the use of cables within the box. This goal was consistent with minimizing cost and maximizing reliability. The use of modular port and management cards led the team to believe that the power supply could also be modular and plug into the backplane in a similar manner. To avoid potential safety hazards, the power supply module is not accessible without opening up the box; however, the interconnection of the supply with the backplane is achieved with the same type of connector used on the logic option modules. The only cable used in the DECconcentrator 500 device provides power to the fans.

Figure 10 is a diagram of the various modules that compose the DECconcentrator 500 hardware.

Power and Packaging Trade-offs

Once we decided that three modules could support 8 to 12 end users, we focused on the selection of packaging. Two basic proposals were examined. The first was to modify the Digital's NAC common box which has been used in many of Digital's products. The second proposal was for a new box design, which allowed improved serviceability via quick access to all FRUs (field-replaceable units). The existing common box design was chosen to

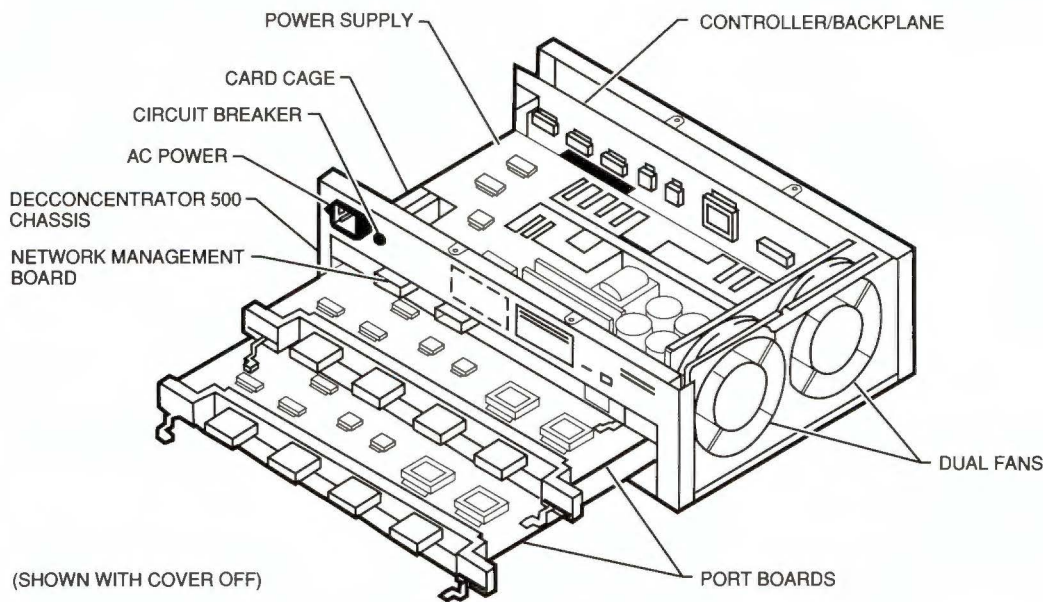


Figure 10 DECconcentrator 500 Hardware Modules

minimize risk to the product development schedule. The necessary modifications were the addition of a card cage for the port and management modules, provision for mounting the power supply and fans, and improvement of the airflow characteristics.

Cooling was also seen as a potential problem in product development. The bipolar logic used in the FDDI physical medium dependent layer (PMD) dissipates a considerable amount of power in a small area. Evaluation of the NAC common box showed that the grill area in each side had approximately 35 percent open area. Analysis showed that significant improvements in airflow could be achieved by increasing this percentage. Mechanical rigidity was traded off against airflow improvement to reach a compromise of 50 percent open area in each side of the box. When prototypes were tested, we found the improvements in cooling followed predictions. The modification also yielded considerable reduction in acoustic noise levels, which allowed the use of off-the-shelf ball bearing fans with good reliability.

Mechanical and electrical requirements could not be met by any of Digital's existing power supplies. The power supply height was limited; a unique connector was required to interface to the backplane; and the available area was determined by the size of the card cage on top of which the supply was mounted. Electrically, the supply had to provide a relatively large amount of -5.2-volt power to

support the emitter-coupled logic (ECL) in the FDDI PMD. Fortunately the total power requirement was similar to that of Digital's Ethernet-to-Ethernet bridge product, the LAN Bridge 200. The power supply group agreed to modify this existing high-volume supply to meet the requirements of the DECconcentrator 500 product. The use of an existing design was expected to result in fewer bugs, and this proved to be the case. Only one bug was found in system stress testing, and it was easily corrected with a minor design change.

Card Handles

The module handle design was probably the single most complex part of the mechanical design, and it had the potential to impact the cost of the product. The mechanical design team recommended modifying a handle design from an existing Digital product. This essentially meant stretching the handle and making openings as required for I/O connectors and LED displays. Handles for both DECbridge 500 and DECconcentrator 500 products are processed from the same mold since only I/O connector and switch/indicator openings are different for the two products.

Traditionally, these handles have been made from a plated cast alloy. Plastic offered the potential of significant cost savings and weight reduction. However, there was concern about the quality of plating with plastic, as well as about the structural strength. The decision was made to start the development

effort for plastic handles while using machined metal parts for the interim. When the final plastic product was available, it met all requirements.

Another critical factor in handle design was electromagnetic interference (EMI). Each FDDI duplex connector required a large, 1.4 cm by 3.8 cm, opening in the handle which posed the risk of emitting radiation. A "waveguide-beyond-cutoff" structure was evaluated. This structure is a rectangular extension to the handle with an opening the size of the FDDI connector. The design attenuates all emissions below the cutoff frequency and was predicted to provide excellent attenuation performance for all harmonics of the FDDI signals. Testing of prototypes verified the emissions problem created by the connector opening and proved the effectiveness of the waveguide structure in eliminating the emissions. This structure was then included in the design of the handles for each FDDI port.

Logic Design

The logic design team developed prototype hardware as quickly as possible so that the diagnostic, operational firmware, and common node software (CNS) firmware teams could proceed with hardware-based debugging. First-pass prototypes of the controller/backplane module and the four-port module were in the laboratory within six months of the start of the project. The GenRad HILO simulation software was used in the module design process.

One type of bug was discovered in the first-pass prototypes that was not caught in simulation. A through-hole component body was used in schematic capture instead of a surface mount body. As a result, the layout was wired according to the through-hole pinout. This error was not caught by any of the software that checks design rules. Thus, the controller/backplane modules required engineering change order (ECO) wires to mount a component onto the back of the module in a "dead bug" fashion.

On the four-port module, the differential ECL signal detect lines from the fiber-optic receiver to the clock and data conversion receiver (CDCR) component were crossed. This logic was not included in the simulation due to its analog nature. The problem in this case was an inconsistency in nomenclature between the CDCR and fiber-optic receiver chip bodies used in schematic capture.

The strategy of building first-pass hardware prototypes as quickly as possible to support early firmware debugging was successful. Simulation played a key role in guaranteeing functionally correct

designs. When second-pass prototypes were tested, only a single ECO wire was required in the product, and the product shipped with the second-pass designs on all modules. The following sections examine several areas of the logic design that are unique to the concentrator function.

Concentrator Port An FDDI concentrator consists of a group of serially connected ports, each of which implements the FDDI PHY functionality. Key to our product was Digital's PHY chip, which implements the physical layer functionality and supports the physical connection management requirements for station management. In addition to the PHY chip, the CDC chips (one each for transmit and receive) provide the serial/parallel conversion, clock recovery on the receive side, and nonreturn to zero inverted (NRZI) encoding/decoding. The fiber-optic transceiver, however, had to be purchased from an outside vendor. Since many mechanically incompatible devices are on the market, we tested the products of many vendors. Fortunately for the product development teams, our optics group was able to identify pin-compatible transceivers from two vendors. Dual sourcing of the transceivers used in our concentrator ports reduced the risk of shipping products based on this relatively new fiber-optic technology.

Internal Token Ring Data Path The FDDI PCM process provides fault coverage and topology rule checking between any two connected FDDI ports. This is essential to ring stability since a token ring is made up of a series of physical connections, any one of which can bring down the entire ring. Because connections between ports within a concentrator are not specified by the FDDI standard, they are a critical design area for ring availability. The DECconcentrator 500 design addressed this aspect of the product as described below.

- Data path (port-to-port) integrity. Adjacent ports in the concentrator interface with a dual-symbol wide data path of 10 bits plus parity for a total of 11 bits. This PHY-to-PHY interconnection is the same interface used to connect PHY to media access control (MAC) in a station. Parity checking was added to Digital's PHY chip to ensure that intermittent or hard faults could easily be detected. If a parity error in this data path occurs, the DECconcentrator 500 device is taken off line to ensure that the entire token ring is not corrupted. Without parity protection, a hard failure on this internal path stops ring traffic altogether, in which case the SMT trace function

might isolate the fault. However, an intermittent fault in a concentrator's internal data path that is not protected with parity could arbitrarily reduce ring performance and increase the risk of undetected data errors and would not be isolated by any of the mechanisms built into SMT.

- **Fault detection/isolation.** The controller/backplane and the PHY chip design allow the DECconcentrator 500 device to offer continuous service in the presence of hardware faults by isolating the faulty hardware from the data path. The diagnostics that are invoked at power up or on command from firmware (as in the SMT trace function) have the ability to isolate faults very close to the component level. The fault report is then passed to initialization firmware which configures the DECconcentrator 500 product so that the faulty hardware is not included in the data path. Two levels of bypassing are provided, one at the port level and one at the option module level. Bypassing is always performed one level of hardware away from the detected fault. Thus if a fault is detected at the CDC component level (using a CDC loopback test), then that single port is bypassed in the PHY chip. If a fault is detected at the PHY level or between PHY chips within a module, the entire module is bypassed on the backplane. Note that power-up diagnostics do not provide a PMD external loopback test except in a special manufacturing mode. Fault coverage of this hardware is provided by the SMT PCM process, which prevents a faulty connection from being included into the ring. Fiber loopback connectors are included with each product for isolating media faults between the fiber-optic transceivers and the fiber-optic cables.
- **Internal MAC attachment.** A MAC is not required for an FDDI concentrator to function, however, it is included as an option to provide remote management. The internal MAC can be thought of as a "management station" attached to one of the concentrator ports whose job is to provide control/status of the local concentrator function. This attachment is internal to the concentrator, but must provide the same basic service as a physical connection to an external station. This service is provided by logic in the data path referred to as the "null PHY." The null PHY provides a means of bypassing the internal MAC if diagnostics should find a fault in any of the hard-

ware [MAC, CAM, ring memory controller (RMC), and packet memory] related to the data link layer. It also provides ring scrubbing in case the MAC should have to leave or enter the ring while the ring is operational.

Upgradeable Nonvolatile Program Memory To support firmware upgrades over the network, the FDDI products require electrically erasable programmable read-only memory (EEPROM). All code in the DECconcentrator 500 product is executed directly out of EEPROM since the microprocessor's clock rate is relatively slow. In the first-pass design we used conventional 32K by 8 dual in-line package (DIP) devices as they were qualified within Digital. In order for the controller/backplane module to accommodate sufficient program memory, we needed a denser technology. At the time, component engineering was evaluating flash EEPROM technology. Flash devices became available in surface mount packages with a density of 128KB that met our needs. The flash memory proved to be a robust technology; however, development of a flash programming algorithm was challenging and required extensive testing. The older EEPROM technology had built-in logic to handle the details of the erasing and programming steps, but with the flash memory these details were directly controlled by software.

In order to upgrade firmware over the network, a well-controlled procedure was developed. A firmware image plus the flash programming code is transferred over the network through multiple packets and stored in packet memory. This down-line upgrade is provided by a network device upgrade (NDU) utility that was developed for the FDDI products. Once the entire image is received in packet memory, it is checked against a cyclic redundancy check (CRC) included in the image. If the CRC is correct and the firmware image is of the correct type (destined for this product), the DECconcentrator 500 product takes itself off line. The 68000 microprocessor then executes the flash programming firmware directly from packet memory to load the new image. Once the load is complete, the firmware forces a reset, and a power-up self-test is run that includes a CRC check of the contents of flash memory.

Software Design Issues

Essential to the completion of the development process was the use of common software and the field testing of the DECconcentrator 500 product.

Common Software Early in the development process it was clear that the aggressive time-to-market goals for the FDDI product set would require the development team to be resourceful. In the beginning of the development cycle a significant effort was made to identify ways to shorten the software development cycle. Code that could potentially be shared among the products, and code that could be ported from previous projects was identified. These early efforts resulted in the common use of the real time operating system (RTOS), common FDDI chip diagnostics, diagnostic error logger, diagnostic dispatcher, and common node software (CNS). In addition to the code that is common among the FDDI product set, much time was saved by porting portions of the remote bridge management software (RBMS) responder and maintenance operation protocol (MOP) from the LAN Bridge 200 product. The management model for the data link and physical layer entities for both the DECbridge 500 and DECconcentrator 500 products was developed to ensure commonality between the two products.

Field Test The field test provided valuable information regarding the quality of the products. Several of the sites chosen to field test the FDDI backbone products were technically knowledgeable about networking. They were able to perform specific testing while monitoring their networks. As a result, detailed test information was provided to engineering. One engineer was assigned to each field test site as a "site parent." The site parents monitored their sites and channeled the information back to engineering. This structure for supporting the field test enabled engineering to react quickly to the needs of the sites as well as act on problems found. This testing and feedback, coupled with the capability to load new firmware revisions over the network, was crucial to achieving quality prior to first customer shipment.

Conclusions

The dual ring of trees topology is well suited for all FDDI environments that require fault tolerance, scalability, and flexibility of configuration. This topology is the right choice for managing the ever-growing local area networks through the 1990s.

Several factors proved to be crucial to meeting both functional and time-to-market requirements with a quality DECconcentrator 500 product.

- Establishing architecture and product feature requirements early and maintaining these with minimal changes.

- Establishing and maintaining a close-knit product development team with good communication channels.
- Leveraging wherever possible from proven and available designs, making incremental improvements as needed.
- Providing a thorough testing process for both hardware and software which tested the product in realistic environments with a process in place to correct problems and verify fixes quickly. We had quite robust products at the time of first customer shipment as a result of our test/fix/verify process.

Acknowledgments

People from many organizations worked together to achieve the development of the product on time. The key members of the engineering design team were Dave Benson, Gerry Capone, Stan Chmielecki, Paul Ciarfella, Alison Coolidge, Janis Roth Cooper, Joe Dagdigian, Tom Ertel, Cheryl Galvin, Dave Hyre, John Iannarone, Bill McCarthy, Charlie McDonald, Dick Muse, Luc Pariseau, Dave Sawyer, and Karen Shay.

References

1. *FDDI Station Management (SMT) Preliminary Draft Proposed American National Standard*, ANSI X3T9/90-078, Rev. 6.2 (May 1990).
2. *Building Wiring Standard for Industrial and Commercial Use*, EIA/TIA 568, PN1907-B (Washington, DC: Electronics Industries Association, Engineering Department, forthcoming 1991).
3. *Detailed Specification for 62.5- μ m Core Diameter/125- μ m Cladding Diameter Class Ia Multimode, Graded Index Optical Waveguide Fibers*, ANSI/EIA/TIA-492AAAA-1989 (Washington, DC: Electronics Industries Association, Engineering Department, 1989).

DECelms—Managing Digital's FDDI and Ethernet Extended Local Area Networks

The DECelms software product provides extended local area network management for Digital's Ethernet/IEEE 802.3 and fiber distributed data interface (FDDI) bridges and for its FDDI wiring concentrator. Product development entailed keeping pace with a changing set of requirements. These included the evolving ANSI FDDI standard, the proposed Digital Network Architecture FDDI data link specification, the Enterprise Management Architecture, the ability to extend the serviceability of the products, and the aggressive schedules of the hardware and firmware development teams. DECelms development resulted in an improved network management functionality including fault, performance, and topology management. These advanced features required corresponding enhancements to the user interface and dependable documentation. The development team met these challenges and successfully delivered the DECelms product to market as a part of Digital's FDDI program.

DECelms Development

The DECelms product, Digital's extended local area network (LAN) management software, provides remote network management for Digital's LAN Bridge 100, LAN Bridge 150, LAN Bridge 200, DECbridge 500, and DECconcentrator 500 products. The remote networks are included in the extended LAN by means of transparent bridges. DECelms functionality includes basic SET and SHOW capabilities, fault management, performance monitoring, FDDI ring mapping, automatic device discovery, and user alarms. The DECelms software runs as an application on a VAX processor running under the VMS operating system.

When Digital embraced the new fiber distributed data interface (FDDI) LAN technology, the role that network management would play in the first product set was unclear. As our understanding of the technology grew, and we recognized the differences between the token ring architecture and the carrier sense multiple access with collision detection (CSMA/CD) protocol, the challenge became obvious. Both technical and business reasons made network management a priority.

On Ethernet/802.3 LANs, network management tools provide the ability to monitor the networks and troubleshoot problems as they arise. The failure

mechanisms of Ethernet/802.3 LANs, such as faulty transceiver taps, continuously transmitting stations, and broadcast storms, are well understood. In addition to network management tools, powerful datascope aids the network manager in recognizing and correcting these problems. Moving into the 100-megabit (Mb) world of the FDDI token ring brought a new set of problems, some of which were understood and others not even imagined. We realized the need to offer a network management solution that was capable of performing the same functions as our Ethernet/802.3 products and had the additional functionality necessary to meet the new challenges of the FDDI technology. Further, no FDDI datascope was available to aid in the development of the FDDI product set. To correct this deficiency, requirements for DECelms functionality called for the software to be not only a good network management solution but a key development tool as well.

Key FDDI Differences

Several key differences between the FDDI and Ethernet technologies determined the specific requirements of the DECelms product. First, the physical topology and the location of stations attached to an FDDI ring play a significant role in

how the ring operates. Each station is active and must participate in connection management to form a working LAN.¹ On Ethernet/802.3 LANs, each station is passive until it wishes to use the network. Thus, an Ethernet/802.3 LAN can operate with stations that do not strictly observe the protocol, since less stringent protocol requirements between stations are necessary to make the LAN work. This difference places a high priority on the ability to manage topology, a functionality that is less significant on Ethernet/802.3 LANs. Requirements to build FDDI ring maps and examine third-party station management (SMT) frame data grew from this priority.

A second difference is the need to manage the FDDI physical layer. This need arose mainly because the DECconcentrator 500 product is a physical layer device and the primary building block of FDDI rings. To manage the FDDI physical layer, a network manager must be able to add or remove FDDI stations from the rings via manipulation of the physical layer ports (PHY ports) of the wiring concentrator. In addition, greater visibility was given into the quality of the physical medium, using the link error monitor, for example.¹

Another difference between the two technologies is the order of magnitude increase in performance of the FDDI ring over Ethernet/802.3. The ability to transparently connect 100-Mb FDDI token rings to 10-Mb Ethernet/802.3 LANs using the DECbridge 500 product greatly influenced network management requirements. Using these plug-and-play bridges makes it easier to create network topologies that can funnel high throughput FDDI traffic onto the lower bandwidth Ethernet/802.1 segments. A good management tool is required to monitor and control these topologies.

A final key difference is simply the need to manage the FDDI data link. The performance of the ring operation must be tuned, and having the capability to modify FDDI media access control (MAC) characteristics such as the valid transmission time or target token rotation time can provide the means to accomplish this tuning. Coupled with the modify operations is a rich set of SHOW capabilities, which gives a detailed view of the FDDI data link behavior.

Extended Management Capabilities

Beyond the technical requirements to provide FDDI network management, DECelms requirements were driven by the need for better network management capabilities in general. Providing an integrated product was a key project goal. Since managing FDDI products included managing bridges, the

program team decided that the DECelms product would supersede the current bridge management product, remote bridge management software (RBMS), and incorporate RBMS functionality as a subset of the DECelms operating features. Thus, DECelms software needed to provide basic SET and SHOW capabilities for the LAN Bridge 100, LAN Bridge 150, and LAN Bridge 200 products, as well as for the new FDDI products, the DECbridge 500 and DECconcentrator 500 devices.

In addition to the ability to set and to show system parameters, DECelms software had requirements to provide automatic fault detection, automatic device discovery, performance monitoring, and FDDI ring mapping. Automatic fault detection would be provided by a built-in polling mechanism and user alarms. Automatic device discovery would be accomplished by listening to system identification announcements broadcast by the devices and registering these devices in the DECelms registry. Performance statistics could be calculated from counters kept by the LAN Bridge 200 and DECbridge 500 products and presented to the user in tabular format. FDDI ring mapping would be accomplished by interpreting the data found in the SMT status information frame (SIF) configuration messages. These functions and their implementations are described in more detail later in this paper.

Development under Time-to-market Constraints

The requirements of the DECelms capabilities were stabilizing. Program decisions were now influenced by the time-to-market constraints. Advanced development work on the DECbridge 500 and DECconcentrator 500 products was progressing, and this work was converted into a full-fledged product development effort. When the network management discussions began, this effort was well underway. Marketing and engineering management clearly communicated to the developers the expectation that Digital would be the first vendor with a complete FDDI solution. This pressure prompted the developers to be very creative in choosing methodologies. As a result, the DECbridge 500 and DECconcentrator 500 teams shaved seven months off the original 18-month schedule. The DECelms team was expected to meet these challenging time constraints.

Reducing the time spent on development required the DECelms team to make many trade-offs. The first trade-off concerned whether the product platform would comply with the Enterprise

Management Architecture (EMA). The FDDI project was concurrent with DECmcc development (Digital's EMA-compliant director), but the FDDI program was three to six months ahead. Given the time constraints, the team decided that the FDDI program could not wait for the DECmcc program to catch up. A point product was the only solution that could be achieved in the short time frame. A second trade-off involved optimizing the development effort. Having no EMA experience, the team used the expertise it had recently gained from working on the RBMS version 2.0 development effort. They made the pragmatic decision to port the available RBMS code in order to meet the time-to-market constraints. The team planned to start with the RBMS code, delete code that was not applicable, and add code to provide the desired new functionality.

But at the same time the RBMS code was being refined and expanded, the DECelms team had to keep the basic RBMS management code operational. The firmware teams needed to use DECelms code to debug their new code, so the DECelms code had to evolve in such a way that new code would replace old and supersede its functionality in a short period of time. This method of code development led to a series of operations akin to "brain transplants," where an interface was drawn in the existing code and new code containing additional functionality was added while the product was kept running. The timing of adding functions had to coincide with the development of the peer functionality in the firmware.

Network Management Architecture

With the product strategy taking shape, some difficult technical decisions concerning the management architecture had to be made. Choosing a network management protocol as well as supporting transport and network layer protocols was a major challenge.

Choosing a Protocol

The team identified three options for the management protocol implementation. The first option was using the common management information protocol (CMIP) layered on top of a paired-down implementation of the Digital Network Architecture (DNA) Phase V protocols. This implementation would include a subset of the DECnet Phase V session, transport, routing, and data link protocols used solely for implementing management agents in server products. This option was the most pure

architectural solution and fit in well with Digital's long-term network strategy. But the effort was just beginning as an advanced development project, and the Phase V protocol specifications were still in the review process. The memory constraints imposed by the DECbridge 500 and DECconcentrator 500 devices coupled with the risk that the product might not meet the time-to-market constraints caused this option to be ruled out as a solution.

The second option was to use the CMIP protocol layered over a subset of the DNA Phase IV protocols. This subset is a streamlined implementation of the network services protocol (NSP) over an 802.3 data link. The network layer is null, so the protocol is limited to the extended LAN. This solution was small in size as dictated by the hardware, offered guaranteed end-to-end delivery service, and would take less time to implement than the DNA Phase V option; but there were several drawbacks. The bridge and wiring concentrator management entities were not defined in terms of the EMA entity model, as was necessary to define the CMIP protocol structure. Using the Phase IV transport protocol option did not bring the management architecture closer to the open systems interconnect (OSI) model defined by DECnet/OSI Phase V, and, in fact, would result in "throwaway" transport and network layer code. Thus, this option was not suitable.

The third option was to extend the bridge management architecture and the RBMS protocol to include support for the FDDI products. In terms of pure architecture, this was the poorest solution, since it would merely result in the extension of an already limited protocol architecture. RBMS is layered over the Ethernet/802.3 data link, has a null network layer, and is therefore constrained to the extended LAN. The RBMS transport is connectionless and, thus, does not offer guaranteed end-to-end delivery service. In addition, no asynchronous message support exists, so the delivery of events or traps is unsupported. But RBMS is simple, being based on the IEEE 802.1 standard for network management, and easy to extend. Also, the LAN Bridge 200 development effort produced a new implementation of the management agent that could be ported into the DECbridge 500 and DECconcentrator 500 products. Further, there was the opportunity to port code from the RBMS version 2.0 product. Given the time constraints under which the DECelms team had to operate, the RBMS protocol was clearly the best possible choice.

SMT as Compared to Management over the Protocol Stack

Another approach to network management was via the FDDI station management frames defined in the American National Standards Institute (ANSI) X3T9.5 FDDI working group draft standard version 5.1. To provide this alternative meant adding complexity to SMT that belonged in a more robust management protocol such as CMIP. Further, the SMT standard was unstable in this area; many vendors participating in the standards work were advocating differing FDDI functionalities for SMT, thus, trying to extend SMT beyond its originally intended scope. Digital's position was that network management should be performed using an OSI model where management is an application that runs on top of the protocol stack and is widely available. SMT frames are below the MAC level and, therefore, do not traverse a LAN beyond a local FDDI ring. Thus, management using SMT frames must come from a local station on the FDDI ring. The data provided by the SMT frames, such as the SIF, is valuable to the network manager. The appropriate mechanism to communicate this data is a management protocol using an agent that can communicate across extended LANs. Once again, RBMS could perform this task.

Management Model Definition

After the protocol issue was settled, the next step was to define the FDDI manageable entities and attributes. The DECelms program team had decided to use RBMS, but the chosen long-term strategy was to use CMIP with the EMA guidelines. The RBMS decision affected only the server products and not the FDDI adapters that were also under development. The adapters would be managed via DECnet/OSI Phase V on both the VMS and the ULTRIX operating systems. The RBMS effort could not derail the long-term management strategy of EMA and the migration to OSI. Thus, there needed to be two management structures that offered similar management capabilities but used different mechanisms. Later, a third management structure driven by the internet community would become important, namely, the management information base (MIB) supported by the simple network management protocol (SNMP).

A series of FDDI network management meetings took place involving the FDDI implementors, the DNA architects, and members of the ANSI FDDI standards working group. The goal was to converge on a set of manageable entities and attributes for the

DECbridge 500 and DECconcentrator 500 products, while keeping an eye to the future of EMA. Several key sources of information were available. One source was the existing bridge management architecture, which defined how to manage an Ethernet/802.3 bridge such as the LAN Bridge 200 device. Another information source was the emerging ANSI X3T9.5 FDDI working group draft standard, specifically the chapter concerning FDDI station management. Additional information was found in an early draft of the DNA FDDI data link specification as well as in the approved version of the DNA CSMA/CD data link specification. The final set of inputs came from the product requirements of the DECbridge 500 and DECconcentrator 500 devices. These requirements called attention to management capabilities that went beyond bridging, wiring concentrators, FDDI data links, and Ethernet data links including management of the down-line-load upgrade feature and availability of field-replaceable unit (FRU) status.²³

All of the above data sources had to be assimilated expeditiously because the chip designs were nearing final form. If the infrastructure necessary to allow for the extension of management functionality was not identified, either proposed features would be dropped from the products, or the chips would have to be respun. Keep in mind that the strategy was to have two management structures in place: the first, based on the bridge management architecture to be used for the DECbridge 500 and DECconcentrator 500 products and managed using the DECelms product; and, the second, incorporating the DNA FDDI data link architecture and a future bridge and concentrator management architecture based on the EMA. Design decisions regarding network management entities and attributes were made accordingly and are described below.

The following discussion presents the manner in which the combined model for the DECbridge 500 and DECconcentrator 500 products was developed. The proposed management model for the concentrator was flattened out and inserted into the current bridge architecture. The bridge architecture was extended to include the FDDI data link in addition to the Ethernet/802.3 data link. Additionally, the PHY port entity was added, but instead of being subordinate to the data link as in the management model for the bridge alone, the PHY port entity was at a peer level, as shown in Figure 1. Additional management attributes were defined to bring the visibility into the box as required for each product. This phase of designing a network management

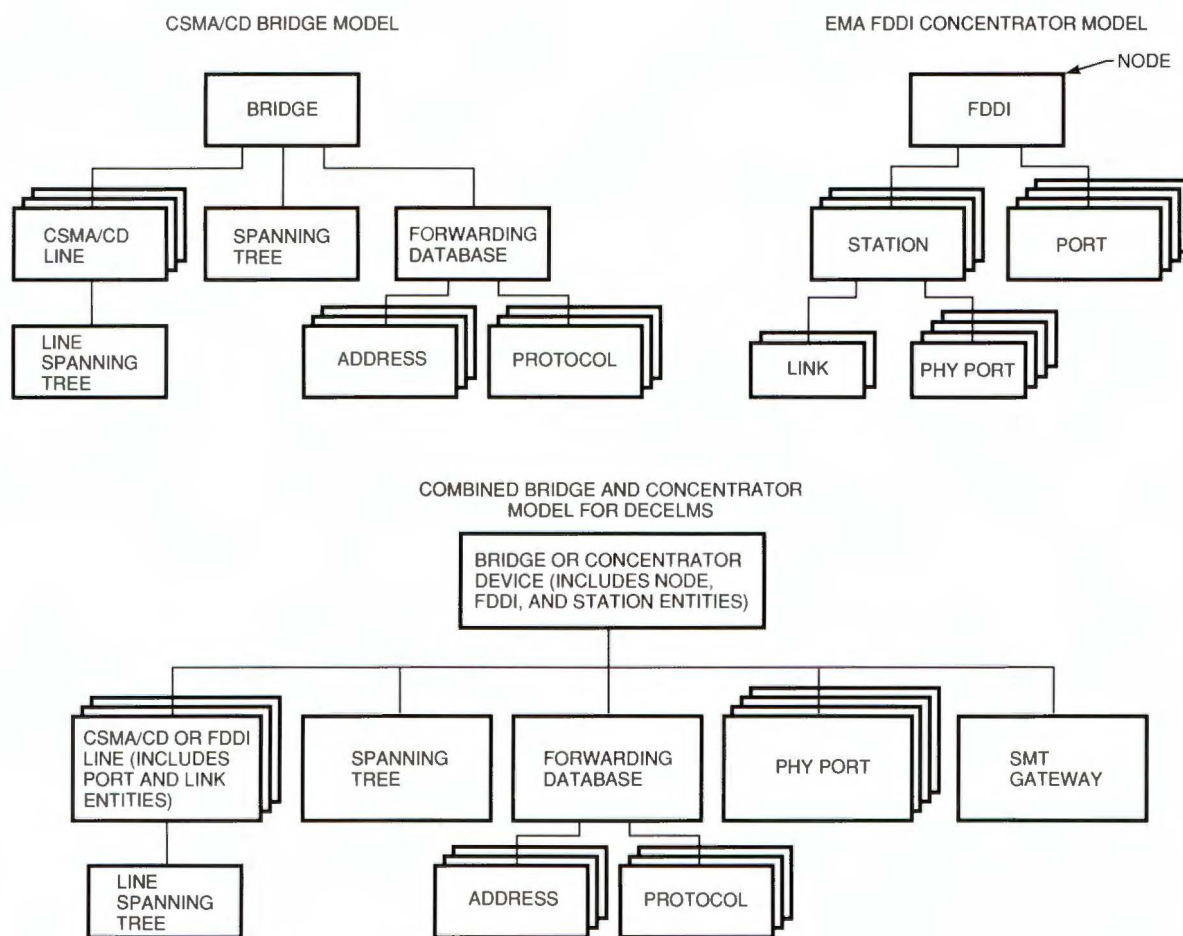


Figure 1 Management Models for the DECbridge 500 and DECconcentrator 500 Products

architecture resulted in a delay in the development of the formal DNA architecture, pending the results of the continuing ANSI standards meetings.

Station Management Gateway

As mentioned earlier in this paper, SMT frames do not travel beyond a local FDDI ring. The DECelms product was expected to manage Digital and third-party FDDI stations on the FDDI ring using SMT frames. This management had to be initiated from the Ethernet/802.3 segment on which the DECelms host was located. The concept of the SMT gateway emerged from this situation.

An SMT gateway is firmware residing on the DECconcentrator 500 device (and subsequently on the DECbridge 500 product) that encapsulates SMT frames into RBMS protocol data units (PDUs) and forwards these frames across the extended LAN. The

architecture that evolved to implement the SMT gateway was another extension of the RBMS protocol. A new entity type called the SMT gateway was added. Gateway requests and responses relating to SIFs, neighborhood information frames (NIFs), and echo frames were defined, for example, GET SIF, GET NIF, and DO ECHO. A set of timers was defined to allow the RBMS host, the SMT gateway, and the target station to be synchronized and to avoid an ambiguous response. For example, if a response is missing, it is important to know if the lack of response is caused by gateway congestion, by no response from the target station, or simply by datagram loss on the Ethernet. With the SMT gateway architecture in place, a DECelms management station located anywhere on the extended LAN can gather SMT frame data from any FDDI SMT version 5.1-compliant station on the same ring as a Digital SMT gateway station.

Management beyond Set and Show

Several management functions beyond basic SET and SHOW capabilities were introduced in an earlier section, namely automatic fault detection, automatic device discovery, performance monitoring, and FDDI ring mapping. These features add value to the network manager. They present network fault and topology data in a timely and concise manner which frees the network manager from interpreting the microlevel details of the network. A primary goal of the DECelms team was to develop a set of tools that the network manager could use to detect and correct a network problem before it was reported by a distressed user.

Automatic Device Discovery

The addition of a station to an Ethernet/802.3 LAN is now a common and simple procedure. Using the plug-and-play nature of Digital's bridge products, the connection of Ethernet/802.3 or FDDI LANs is also straightforward. The addition of stations to an FDDI ring is easy with the use of wiring concentrators. In campus environments, network managers can typically, but not always, control physical access to the network backbone but have less control once a segment enters an individual department. Consequently, there is a need to automatically discover the "renegade" devices that can be added to such networks. The DECelms product solves this problem by providing a device listener that listens to the maintenance operation protocol (MOP) system identification messages periodically broadcast by all of Digital's bridges and wiring concentrators. When a new station is heard, the DECelms software queries the station via the RBMS protocol to obtain more detailed information about the station and automatically adds the station to the DECelms registry. The software also produces a user alarm that notifies the network manager of the existence of this device and allows the manager to evaluate its impact on the topology and performance of the LAN. For instance, every bridge added between two stations adds latency to the protocol communications between the devices. Too much latency may negatively impact LAN performance by causing certain protocols to fail.

Background Poller

The ability to quickly recognize changes in the state or counters of a network device can help a network manager avoid a degradation of service. The DECelms background poller provides this type of fault recognition by keeping information about

the state of each device in the DECelms registry and reporting changes in that state to the network manager through user alarms and a log file. The information includes the designated bridge on a LAN, the number of FDDI ring initializations, the number of cyclic redundancy check (CRC) errors on a given data link, and, perhaps most importantly, the fact that a station has become unreachable.

Network Interface Multiplexer Process

Both the device listener and the background poller are integral parts of the DECelms process known as the network interface multiplexer (NIMUX). As implied by its name, NIMUX also provides the basic multiplexing of user protocol messages as they are sent and received by the data link driver. Designing NIMUX to incorporate these three distinct functions was a considerable challenge. The design includes a mailbox interface to the variable number of user interface tasks. This interface provides for both the sending and receiving of user data and alarms. The NIMUX design also provides the control interface for the background poller and the device listener and a service interface for the DECelms registry, which is only written by NIMUX for synchronization purposes.

Within NIMUX is a kernel that synchronizes these three basic functions. Synchronization is performed using VMS asynchronous system traps (ASTs) and event flags. When the NIMUX has no tasks, the process hibernates to conserve the CPU utilization. But, the NIMUX process may be woken up, for example, by the delivery of a mailbox message from a user process or by the completion of a network I/O operation.

The implementation of the background poller is complex. The background poller gathers state information for each device in the DECelms registry and continually circulates through the registry. This initial query is necessary to determine whether the device is currently active or inactive. To obtain the complete state of a device, the DECelms software must issue multiple protocol requests to that device. In the case of a DECconcentrator 500 device, twelve requests must be issued to accumulate all the device data. The synchronization of these requests is a challenge, since it is possible for any of these requests to fail because of a datagram loss or the failure of the network or the device itself. Several event flags are required to identify when a query is pending and then to identify whether the query is a success or a failure. When a single query is completed, the next step is dependent upon the device type and the status of the previous query.

The automatic addition of a device to the DECelms registry is performed in the polling cycle. Once a device MOP system identification message is heard and the device is recognized as new to DECelms, the responsibility of performing the RBMS protocol queries belongs to the poller. Three protocol messages are necessary to determine the device type and MAC addresses. The recognition that these two functions have this common thread helped to simplify the NIMUX design and limited the amount of internal state information that it was necessary to keep.

In parallel to the background device queries and the automatic device registration taking place, NIMUX also has to process user requests. These requests are given priority over those of the background poller, since an interactive user cannot be made to wait until all the background polling queries are completed. This processing is accomplished in the outermost loop of NIMUX. If a user request is delivered via a mailbox, an event flag indicates its arrival. This request is given preference over moving on to the next step in the polling process.

An important part of the interface between NIMUX and the user processes is the ability to deliver alarm messages. Each user process has the option of enabling or disabling alarm messages for its process. The fact that multiple alarms can be generated by a single device query from the background poller, and multiple users need to receive these alarms, means the interface has to contain queues to buffer the alarms and state variables to control the sending of the data. This problem was solved, as were most of the NIMUX interface problems, by the addition of these data structures to the bridge control process (BCP) tables. A BCP table is a robust data structure that keeps the NIMUX state information for a single user process. One table is created and destroyed for each user process that exists.

The last job NIMUX can perform is giving write access to the DECelms registry. Since both the automatic device registration function in NIMUX and a user process can add devices into the registry, synchronization is necessary. The simplest solution is to eliminate the need to synchronize by having only NIMUX do the registry write operations. The mailbox interface commands were extended to allow user registration, modification, and deletion of devices in the DECelms database using NIMUX as the server.

FDDI Ring Mapping

The ability to map the FDDI ring is a key feature of the DECelms product. The SMT gateway provides the mechanism to obtain the raw SMT frame data necessary to build ring maps. Two types of ring maps were identified as possible functionalities: a simple logical ring map and a physical ring map. The logical ring map could be built using the upstream neighbor address (UNA) contained in the NIFs. This functionality would provide a list of the MAC addresses in the FDDI ring to which the token was passed, but would not provide information regarding the number of PHY ports in a concentrator or the details of the physical topology of the network.

The preferred functionality was the physical ring map which can provide detail into the actual physical topology of the FDDI ring. Using the theory of wiring concentrators to build FDDI rings, the resulting topology is a ring of trees. Typically, this configuration includes dual attachment concentrators (DACs) located on the dual FDDI ring, with other concentrators and single attachment stations (SASs) connected to the concentrator on the dual ring in a tree-like fashion.³ The physical ring map is required to represent this topology showing the PHY port attachments in addition to the MAC attachments.

This representation is accomplished using the data contained in the SIF configuration messages. The physical ring mapping starts when the user supplies the address of an SMT gateway on the target FDDI ring. The DECelms host then issues GET SIF gateway requests to the gateway. The algorithm calls for the ring map to start with the gateway's SIF information. Then, the UNA of the gateway contained in the SIF configuration message indicates which station in the ring to query next. Using the station descriptor, station state, and path descriptor fields of the SIF response, the DECelms host can derive the detailed physical map.

The basic mechanism to provide the physical ring map was understood, but the implementation and the actual application mandated additional requirements. With the SMT standard still in a state of flux, testing the ring map with third-party vendors disclosed that it was possible to interpret the standard in different ways. Consequently, the ring map code had to be more flexible to interpret the same data fields which often had slight variations in format due to loose interpretations of the specification. With increased testing the ring map could interpret data from many vendors and thus became a very useful and popular tool.

Another feature was added to the ring map which made it possible to build partial ring maps. The capability of specifying the ring map starting address allows a user to pass a station that cannot be mapped otherwise (i.e., the station does not correctly respond to a SIF request) and manually continue the map with the next station on the ring.

The display of the map was a noteworthy task. Concepts for this implementation ranged from a simple tabular format to color graphics. The time constraints greatly influenced our decision. A simple and concise set of icons was developed to represent the stations on the ring. These icons are actually depicted as stick figures displayed using ASCII terminal art. Each icon is paired with the level at which it exists in a tree, similar to the way in which logic statements appear in a compiler listing. Figure 2 is an example of a ring map display. This icon implementation was achieved in far less time than a graphics display solution would have taken and offers the user a picture of the FDDI ring map,

which is an advantage over simply presenting a table. In actual use, the map is easy to understand and provides an intuitive picture of the network topology.



User Interface





As described earlier, the DECelms code has its roots in the RBMS software. However, the RBMS user interface is a simple command line which lacks a number of features necessary to make the product more useful to network managers. To remedy this deficiency, the DECelms team set the following goals for the user interface. The software needed to provide intuitive commands to a previous RBMS user, shorter commands than RBMS, the ability to scroll through long output displays, the ability to input data from command files, and the ability to redirect output into files. During the process of reaching these goals, the product had to be continuously available as a development tool.










A screen-based, user interface solution, known as the DECelms screen manager (ESM), was designed.

```

FDDI Ring Map                                     As of: 14-NOV-1990 09:52:48
Name: Defcn                                       Address: 08-00-2B-13-E0-BB

Concentrator:  Station:  P - Primary Ring
                                                    S - Secondary Ring

Attachments: trunk == b  a  s 

-----
Ring : S ^ P      Station      Station Name      Station ID
v   |      Icon          Type      MAC Address (MAC#)
|   |              Active (concentrator) ports
:   |
:   | +== DEFNCN          ID=00-00-08-00-2B-13-E0-BB
:   | |      8-port 1-mac DAC      08-00-2B-13-E0-BB
:   | |      Active: 3, 4, 5, 6
:   | |
:   | | +-- 1 BRIDGE1          ID=00-00-08-00-2B-14-D3-29
:   | | |      SAS                  08-00-2B-14-D3-29
:   | | |
:   | | | +-- 1 WRKSTATION      ID=00-00-08-00-2B-14-D3-87
:   | | | |      SAS                  08-00-2B-14-D3-87
:   | | | |
:   | | | | +-- 1 BRIDGE2          ID=00-00-08-00-2B-14-D0-BA
:   | | | | |      SAS                  08-00-2B-14-D0-BA
:   | | | | |
:   | | | | | +-- 1 BRIDGE3          ID=00-00-08-00-2B-14-D0-C1
:   | | | | | |      SAS                  08-00-2B-14-D0-C1
:   | | | | | |
:   | | | | | | +== DEFNC2          ID=00-00-08-00-2B-14-13-9E
:   | | | | | | |      8-port 1-mac DAC      08-00-2B-14-13-9E
:   | | | | | | |      Active: 3, 4, 5
:   | | | | | | |
:   | | | | | | | +-- 1 BLD1BRIDGE      ID=00-00-08-00-2B-14-D3-23
:   | | | | | | | |      SAS                  08-00-2B-14-D3-23
:   | | | | | | | |
:   | | | | | | | | +-- 1 3RDPARTYSAS      ID=00-00-08-00-2B-14-D3-24
:   | | | | | | | | |      SAS                  08-00-2B-14-D3-24
:   | | | | | | | | |
:   | | | | | | | | | +-- 1 DEFEB          ID=00-00-08-00-2B-14-D0-C0
:   | | | | | | | | | |      SAS                  08-00-2B-14-D0-C0
:   | | | | | | | | | |

---- End Of Map ----

Note that attachments a, b, and s are not illustrated in this example.

```

Figure 2 FDDI Ring Map

ESM uses VMS screen management (SMG) to provide the basic screen manipulation. Two screen modes were developed: an input mode and an output mode. In the input screen mode, commands are echoed and error messages displayed. This mode provides command line recall, command line editing, and support for command files. The output screen mode is used to display the data from command requests. This mode provides the user with scrolling driven by the keypad and options to move the output data into files. The integration of ESM into the DECelms software was accomplished through a well-defined interface that could take the output of the original RBMS output and convert it into the required format for ESM. One by one, the old screens were replaced by ESM, and at any given time during development there was a working user interface that provided debug support for the firmware developers.

Summary

This paper presents two important themes. The first concerns the technical challenges and accomplishments of the DECelms product. Among the challenges were defining the network management architecture, including the protocol, the manageable entities and attributes, and the SMT gateway. Technical accomplishments included the design of NIMUX and its multifaceted functionality. The second theme was how the development team proceeded to build the product. The time to market became the controlling factor in many product decisions. Trade-offs were made in favor of product functionality and meeting users' expectations rather than to promote and preserve architectural and design purity. Above all, product quality remained the top

priority and the motivation for the DECelms development team to strive for excellence.

Acknowledgments

The DECelms product was a significant accomplishment, completed in a short period of time. It was the result a great team effort and personal sacrifice by the DECelms development team. I want to recognize the following individuals for their efforts as members of the engineering team: Barry Colella, Mary Ellen Monette, Phil Morano, Ken Pruyn, and Herman Levenson. In addition, I want to thank Andrew Shores for his diligence and endurance in producing the DECelms user documentation. Lastly, I want to thank members of the FDDI program team, the DECbridge 500 and DECconcentrator 500 teams, and the NAC architecture group for their support and contributions that made the DECelms product a success story.

References

1. P. Ciarfella, D. Benson, and D. Sawyer, "An Overview of the Common Node Software," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 42-52.
2. R. Kochem, J. Hiscock, and B. Mayo, "Development of the DECbridge 500 Product," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 53-63.
3. W. Tiffany, G. Koning, and J. Kuenzel, "The DECconcentrator 500 Product," *Digital Technical Journal*, vol. 3, no. 2 (Spring 1991, this issue): 64-75.

ULTRIX Fiber Distributed Data Interface Networking Subsystem Implementation

The ULTRIX operating system, Digital's version of the UNIX operating system, supports the first implementation of a host networking subsystem with a fiber distributed data interface (FDDI) network adapter. Digital's FDDIcontroller 700 adapter provides a single FDDI attachment for the reduced instruction set computer (RISC)-based, DECstation 5000 model 200 platform. Combined with the ULTRIX networking subsystem, this adapter brings high-speed communication directly to the workstation.

Digital made the decision to adopt fiber distributed data interface (FDDI) local area network (LAN) technology to follow Ethernet. With the FDDI system, Digital is developing products to support improved network performance such as the high-speed interconnection of workstations.

The ULTRIX operating system supports Digital's first implementation of an FDDI host networking subsystem. A key decision in the ULTRIX FDDI program was to design an adapter for reduced instruction set computer (RISC)-based workstations. Consequently, the DEC FDDIcontroller 700 network adapter was designed to support an FDDI single attachment for the DECstation 5000 model 200, RISC-based workstation. This support covers the Defense Advanced Research Projects Agency (DARPA) internet network protocols designed for the ARPANET packet-switched network. The DARPA internet network protocols include the internet protocol (IP), the transmission control protocol (TCP), and the user datagram protocol (UDP).

This paper begins with an overview of the ULTRIX operating system. The sections that follow present the implementation details of the network and communication driver, review specific issues in the ULTRIX FDDI implementation, and discuss both performance and future directions.

Overview of the ULTRIX Operating System

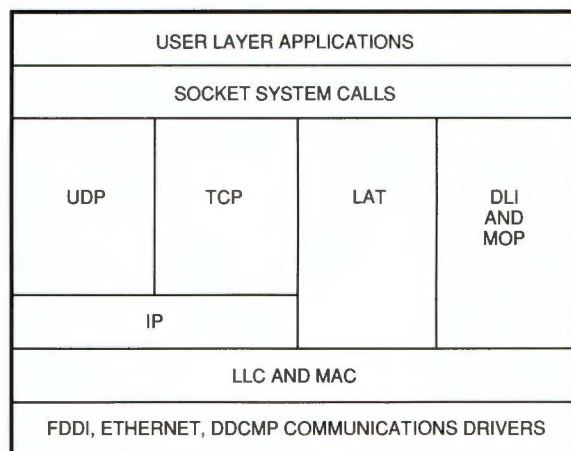
The ULTRIX operating system is based on the 4.3 BSD system. (BSD refers to Berkeley Software Distribution, a popular version of the UNIX operating system.) As in other systems based on the UNIX

system, the ULTRIX operating system operates in user and kernel modes. A process running in user mode can be preempted. Interrupts are run in the context of the current process. A process running in kernel mode voluntarily relinquishes control of the CPU. ULTRIX networks and communications device drivers run in kernel mode.

The ULTRIX operating system supports network activity through a well-defined, layered hierarchy including user applications, system calls, and compile-time entry points to the protocols and communication device drivers. The layered hierarchy is illustrated in Figure 1.

The user layer consists of applications (e.g., electronic mail) that use specific system calls to support network activity. These interprocess-communication system calls incorporate the notion of a socket and, hence, are referred to as socket system calls. Sockets are endpoints of communication containing information used by the operating system to associate data with specific clients and servers. When executing in kernel mode, the socket system calls perform the memory management, the security checking, and the state management common to all protocols. When the protocol-common processing is complete, the operating system accesses a protocol switch table containing vectors to protocol-specific modules. These modules, in turn, access communications drivers through the network interface table.

The ULTRIX environment is characterized by a large number of servers, i.e., SUN's NFS system, which allows remote access to entire file systems, and many network applications. The servers support a



KEY:

UDP	USER DATAGRAM PROTOCOL
TCP	TRANSMISSION CONTROL PROTOCOL
IP	INTERNET PROTOCOL
LAT	LOCAL AREA TRANSPORT
DLI	DATA LINK INTERFACE
MOP	MAINTENANCE OPERATION PROTOCOL
LLC	LOGICAL LINK CONTROL
MAC	MEDIA ACCESS CONTROL
DDCMP	DIGITAL DATA COMMUNICATION MESSAGE PROTOCOL

Figure 1 ULTRIX Network Subsystem Layering

diverse range of activities such as managing mail and ensuring that X Window System managers are available to remote workstations. The underlying protocols for most of these servers are TCP, IP, and UDP.

ULTRIX Support for the FDDI System—Development Strategies and Issues

Presentations by Digital's networking architects in May 1988 brought the earliest news that Digital was pursuing a timed, token ring topology (i.e., FDDI) in contrast to Ethernet, which employs a carrier sense multiple access with collision detection (CSMA/CD) data link protocol. Digital's FDDI engineering program began with product requirements for a wiring concentrator, a bridge to link Ethernet and FDDI networks, and an FDDI adapter to the VAX computer. The FDDI program team planned only high-end system direct connectivity to the ring. Workstations would be connected through the existing Ethernet across a bridge.

However, the ULTRIX operating system running network applications on RISC workstations was already saturating the Ethernet. The ULTRIX engineering group advocated FDDI adapters, not only for RISC-based servers but also for the increasing number of high-end, RISC-based workstations.

ULTRIX and VMS engineering groups began architectural discussions with the FDDI development

groups to write requirements for FDDI adapters for both RISC and VAX processors. Due to the evolving ULTRIX emphasis on RISC-based solutions, the ULTRIX engineering group represented data structure, virtual addressing, and performance requirements for RISC processors, while the VMS engineering group represented the same requirements for VAX processors. Approximately ten months after the initial network architecture presentations, the FDDI program team drafted product requirements for the ULTRIX implementation, including support for an FDDI workstation adapter.

To provide a workstation solution, members of the ULTRIX engineering group had already begun to work with the Low End Network Systems (LENS) Group on an advanced development project to define a workstation-based FDDI adapter. The team discussed alternatives for FDDI workstation connections, including the emerging DECstation 5000 model 200 TURBOchannel bus, the DECstation 3100 plug-in option, and the industry-standard small computer systems interface (SCSI) bus.

Six months into the adapter advanced development project, the internet community confirmed interest in TCP/IP implementations for FDDI requirements by issuing a draft of the request for comment, RFC 1103 (recently renamed RFC 1188), which defines the encapsulation of internet packets on FDDI networks. Members of the FDDI engineering team were instrumental in providing direction for the internet FDDI task force meetings on RFC 1103 and the FDDI network management information base (MIB). The draft of RFC 1103 prompted internet vendors to hastily implement FDDI workstation-based products and the LENS group to publish plans for FDDI connectivity to RISC-based workstations with ULTRIX support.

In October, at the Interop '89 Conference in San Jose, California, several internet vendors showcased FDDI products. Although Digital did not show FDDI products at the conference, this event prompted Digital to design an architecturally sound, high-quality, FDDI solution to gain a competitive edge.

Soon after the conference, the FDDI Data Link Specification and the project plan for ULTRIX support for the FDDI system were released. Subsequent ULTRIX development efforts to support the FDDI system produced new networking code for the TURBOchannel device driver, the data link layer, and the network layer. These efforts paralleled the TURBOchannel adapter development efforts.

A prototype ULTRIX implementation successfully passed 802.2 frames over an Ethernet connection,

as required by the American National Standards Institute (ANSI) FDDI standard, to exercise the data link and network layer changes necessary for FDDI support. The product announcement for the TURBOchannel FDDI adapter assigned the official name DEC FDDIcontroller 700 to the adapter. Prototypes were delivered in May 1990; firmware integration was completed; and the first address resolution protocol (ARP) broadcast packet was sent over an FDDI ring from an ULTRIX host.

Device driver and adapter interoperability problems such as timing considerations, data corruption, and performance issues were solved promptly by close cooperation between the software and hardware groups. Several additional performance enhancements were added to the operating system, bringing the performance of the ULTRIX and adapter combination to nearly 40 percent of the entire FDDI bandwidth—a factor of four times greater than existing Ethernet implementations.

At its trade show, DECWORLD 1990, Digital announced the availability of its FDDI product offerings. These included the DECconcentrator 500 and DECbridge 500 products, and the DEC FDDIcontroller 700 adapter, which runs under the ULTRIX operating system.

ULTRIX Internals

The implementation of FDDI support in the ULTRIX operating system required the development of a link-level architecture and a network device driver.

Operating system changes to improve the performance of the network were made later. The next two sections describe the implementation of the link-level architecture and the device driver. Performance changes are discussed in the ULTRIX Network Performance section.

Data Link Support

The ULTRIX operating system implements both the internet protocols (TCP/UDP/IP) and the Digital Network Architecture (DNA) model, including the Digital data link interface (DLI). In both the internet and DNA models, the data link defines services known as the logical link control (LLC) and the media access control (MAC). A major challenge in the implementation of ULTRIX FDDI support was defining a set of common data link routines to satisfy the frame format requirements of both internet and DNA models in a heterogeneous LAN environment.

Prior to the introduction of the FDDI system, all ULTRIX internet networking for LANs ran over Ethernet networks using Ethernet V2 frame formats, even though ULTRIX DLI networking supported both Ethernet V2 and 802.2 LLC frame formats. Figure 2 illustrates the differences among the V2 Ethernet and 802.2 Ethernet and the FDDI frame formats. V2 and 802.2 frames include Ethernet encapsulation; 802.2 frames consist of the MAC, the LLC, and data segments. When the 802.2 frame is sent over the FDDI system, the FDDI framing adds the FDDI-specific encapsulation, as shown in Figure 2.

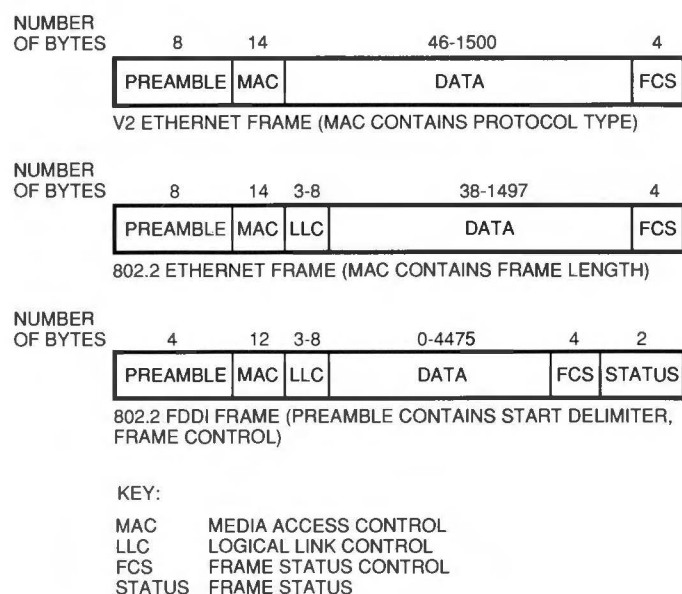


Figure 2 Frame Formats

In order to conform to the ANSI FDDI standards, which require 802.2 LLC frame formatting, members of the Internet Network Working Group wrote Internet RFC 1188.¹ This RFC specifies the rules for 802.2 LLC encapsulation of internet frames on an FDDI network. To meet the needs of both Ethernet and FDDI networks, we designed and integrated a set of network-common routines. These routines, the `net_output()` and the `net_read()` functions, support 802.2 encapsulation as required by RFC 1188.

net_output() Function The `net_output()` function prepares packets for transmission by ULTRIX network communication device drivers. If a driver requires 802.2 LLC support, the `net_output()` function supplies the necessary header, prefixes the MAC header, and enqueues the packet to the appropriate communication driver for transmission.² The function, while supporting 802.2 LLC encapsulation, does not preclude protocol modules from supporting their own LLC formatting. The encapsulation is switch-driven so implementors can add special routines to the switch to either replace or bypass the 802.2 LLC encapsulation.

net_read() Function The `net_read()` function is called by the communication drivers and prepares received packets for delivery to protocol modules. This function first identifies the protocol type from information contained in the MAC and LLC headers, places the packet on the corresponding queue, and finally schedules a software interrupt to alert the appropriate protocol module of the arriving packet.

Communication Driver

The FDDIcontroller 700 adapter connects directly with the DECstation 5000 model 200 TURBOchannel bus. The FDDIcontroller 700 is a 100-megabit (Mb)-per-second, timed, token ring adapter that supports an FDDI single attachment for the DECstation 5000 model 200. The adapter provides a host interface with the following features: a packet memory interface (PMI) gate array for receive direct memory access (DMA) data transfer; a packet memory subsystem that contains one megabyte (MB) of dynamic random-access memory (DRAM) for packet store and forward; and the ability to handle FDDI ring initialization, recovery, and SMT frame processing. (SMT refers to the ANSI-specified FDDI station management.³) The adapter is controlled by a microprocessor and uses Digital's FDDI chip set, which includes ring memory control (RMC), media access control, and the elasticity buffer and physical link manager (ELM). The ULTRIX communication driver interfaces to the adapter's port architecture.

The port architecture defines the mechanisms to transfer FDDI frames and control or status information between the communication driver and the port. The ULTRIX communication driver interfaces to the adapter through six port registers, six port memory rings, the driver data structures, and the driver data buffers, as shown in Figure 3.

Each port register is represented by 16 bits in adapter packet memory. These registers are described in Table 1.

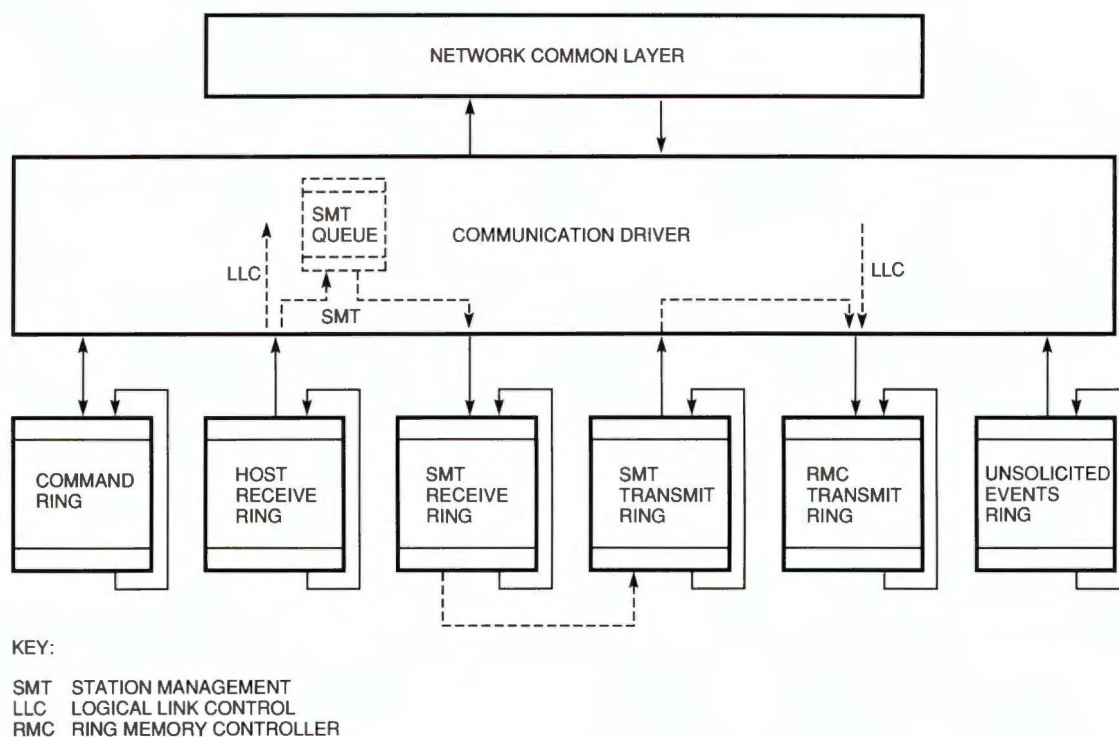
Table 1 Adapter Port Registers

Register Name	Written by	Purpose
Port Reset	Driver	Forces the adapter to reset
Port Control A	Driver	Controls adapter operations
Port Control B	Driver	Indicates that the driver is active
Port Interrupt Event	Adapter	Notifies the driver of important events
Port Status	Adapter	Indicates the adapter status
Port Interrupt Mask	Driver	Masks the adapter interrupt events

The adapter uses six queues, called port memory rings, to exchange data, events, and commands with the driver. These port memory rings, represented as circular lists of descriptors, are described in Table 2. Each descriptor is associated with a data buffer in adapter packet memory or in driver memory.

Table 2 Port Memory Rings

Ring Name	Purpose	Description
Host Receive Ring	Data flow	Identifies driver data buffers to receive incoming packets
RMC Transmit Ring	Data flow	Identifies adapter data buffers containing packets to transmit
SMT Receive Ring	Data flow	Used by the driver to route SMT frames to the adapter for processing
SMT Transmit Ring	Data flow	Used by the driver to route SMT frames to the adapter for processing
Command Ring	Control	Used by the port driver to initialize, configure, and monitor adapter operations
Unsolicited Events Ring	Control	Used by the adapter to notify the driver of unsolicited events



Performance is a key factor in the success of Digital's workstation FDDI offering. A great effort was made to characterize the DECstation 5000 machine performance by using the earlier DECstation 3100 workstation performance results to help set realistic goals. Both the characterization and the measurements were essential to predict the performance goals. This section discusses the level of performance achieved by the DECstation 5000 model 200 running the ULTRIX operating system with FDDI support. We present details of the historical precedence for predicting FDDI performance, the evolution of the ULTRIX networking code, and the performance of applications with the FDDI system.

Historical Precedence

We expect FDDI performance to develop similarly to that of Ethernet. Early Ethernet hosts were unable to utilize more than 20 percent of the available network bandwidth because of the limited throughput capability of existing processors. The graph shown in Figure 4 illustrates the historical performance of several different processors using Ethernet adapters. Note that since 1983, network throughput has increased significantly. At some time after the middle of the decade, it was possible to reach a throughput of 10 Mb per second, a rate high enough to saturate Ethernet with a single host.

Figure 4 also shows that, in nearly all cases, the achievable throughput is limited by the speed of the processor. In addition, an experimental constant of 1 mips/Mb is measured in cases where the processor is saturated. (1 mips equals one million instructions per second.) This constant means that 1 mips of processor speed is needed to generate 1Mb of network traffic. For example, the 1-mips VAX-11/780 processor is able to generate about 1Mb of network traffic. The data presented in Figure 4 shows that a processor follows the 1 mips/Mb ratio unless the adapter becomes a limiting factor, as in the case of the DECstation 3100 system. The 1 mips/Mb ratio allows us to predict that a 100-mips processor is required to saturate the FDDI system. Thus, the FDDI system satisfies the throughput requirements of available processors and allows for the growth of faster processors. Finally, if the present trend of doubling processor speed every two to three years continues, the FDDI graph will resemble the Ethernet graph of the 1980s, with the saturation of the FDDI system possible in 1996 or 1997.

Evolution of the ULTRIX Internet Code

The early implementations of ULTRIX internet networking code were based on robust BSD networking code. In 1987, the ULTRIX internet networking code was updated to incorporate performance enhancements available from a recent BSD release. Later, ULTRIX network performance was further improved to include the implementation of a dynamic buffer allocation policy to replace the inefficient static allocations. With FDDI systems, the challenge then became adapting the code to effectively use the higher network throughput.

We attacked this problem by isolating and optimizing each networking subsystem. The ULTRIX networking code is divided into three major subsystems: sockets, protocols, and drivers. Each of these subsystems can be further subdivided: sockets into the system call interface and the socket-to-protocol interface; protocols into the IP and UDP components; and drivers into the ARP, buffer management, and driver interrupt components. We used kernel profiling, a means for making run-time measurements of time spent in system-level routine calls, and known benchmarks to track progress.

Using an unreliable protocol without error recovery, such as UDP, instead of TCP with reliability and packet sequencing features, the packet-per-second (pps) rate of each component can be easily determined. Figure 5 shows the sizable packet rate measured on the DECstation 5000 model 200 for both the nonoptimized and the optimized ULTRIX network, i.e., before and after performance improvements are incorporated. Note that the pps rate of each layer reflects improvements in the layers below.

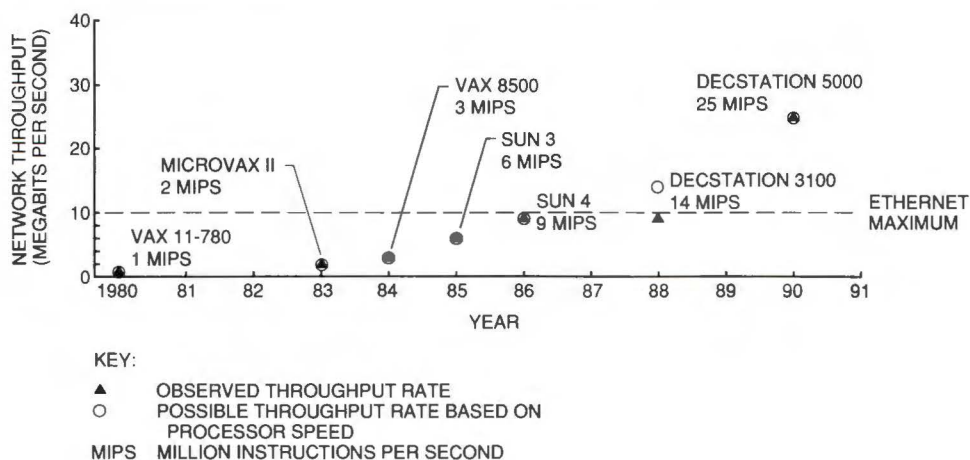


Figure 4 Historical Performance of Processors and Ethernet Adapters

LAYERS OF THE ULTRIX OPERATING SYSTEM	RATE (PACKETS PER SECOND)	
	NONOPTIMIZED	OPTIMIZED
SYSTEM CALL	16000	16000
SOCKET-TO-PROTOCOL	1750	2650
UDP	1700	2570
IP	700	1400
ARP	610	1230
DRIVER START	600	1200
DRIVER DONE	300	600

NOTE: THE DECSTATION 5000 PLATFORM RUNNING UNDER THE ULTRIX OPERATING SYSTEM HAS A 4096-BYTE PACKET-PER-SECOND RATE.

KEY:

UDP USER DATAGRAM PROTOCOL
IP INTERNET PROTOCOL
ARP ADDRESS RESOLUTION PROTOCOL

Figure 5 Optimized and Nonoptimized Packet Rates for ULTRIX Network Components

Packet rate values for the system call through the ARP components are determined by processor speed and code. Rates below the ARP depend on adapter speed and not processor speed. The packet rate value for the ARP is the maximum packet rate for a processor. With an optimized packet rate, Figure 5 shows a maximum rate of 1230 pps for the DECstation 5000 workstation. Since each test packet contains 4096 bytes, this rate is equivalent to 40Mb per second, which is a 40 percent FDDI bandwidth utilization.

A significant amount of work is performed at the socket-to-protocol, IP, and driver start layers because in each case, an effective copy of the data is performed. The socket layer copies data from the user

into the kernel, the IP layer checksums the data, and the driver start routine copies the data to the adapter. We focused our efforts on the socket-to-protocol layer and found that considerable processor time was spent allocating kernel buffers to hold the data. Reworking this code to buffer the data more efficiently resulted in the performance change between the optimized and nonoptimized columns shown in Figure 5.

Application of the FDDI System

The greatest long-term benefit of end-node access to FDDI will probably come to those utilizing a distributed computing environment since this paradigm relies heavily on the performance of the underlying network. While Ethernet currently serves this growing set of applications well, it is expected that as the application demand for network service increases, so will the total network bandwidth and performance requirements of the participating end node.

Even today, some users of distributed network file systems (e.g., NFS) will notice a significant performance improvement as a direct result of using FDDI. This improvement is particularly evident on cached file reads and writes, where no disk access is required but the aggregate bandwidth advantage of FDDI is beneficial. However, overall NFS performance is currently limited by CPU speed and disk write capabilities; so we expect that with improvements in processor performance, disk access, and data caching, network performance at the level provided by FDDI will soon be essential.

Table 3 contains FDDI performance measurements with respect to Ethernet for various applications and transports at the application layer. The spray application is most often used to measure how an unreliable transport performs. Applications such as the BSD rcp (the remote file copy program over TCP/IP protocols), the file transfer protocol (FTP),

Table 3 Application Performance in Relation to Ethernet

Transport	Application	Rate (Megabits) (UDP Checksum On)	Rate (Megabits) (UDP Checksum Off)	Change
TCP	rcp	18	18	2X
TCP	FTP	5	5	1X
UDP/NFS	NFS Read	20	30	2X (3X)
UDP	Spray	22	35	2.5X (3.3X)
TCP	TTCP	18	18	2X
UDP	TTCP	22	38	2.5X (4X)

and the test TCP (TTCP) program all measure performance of a reliable transport protocol. An NFS test is used to measure how FDDI performs as a file server. Note that, in all cases except FTP, performance improves by at least a factor of two. FTP does not take advantage of the larger buffering gained by using the FDDI system and, thus, shows no performance change over Ethernet.

Another aspect of network performance is the routing function. Using the DEC FDDIcontroller 700 adapter, the DECstation 5000 model 200 can perform FDDI-to-FDDI routing, FDDI-to-Ethernet routing, or both, for internet traffic. With a built-in Ethernet port and the ability to accept up to three additional TURBOchannel adapters, a 5000 model 200 can connect to as many as four different networks.

The performance of such a host-based router is difficult to characterize. A wide range of factors influences this performance, including the protocols routed, the efficiency of the routing algorithms, the system load, and the available data link bandwidth. Nonetheless, it is useful to consider the performance of the 5000 model 200 serving as an FDDI router because we expect this product feature to be popular. Table 4 shows the performance results for a DECstation 5000 workstation performing FDDI-to-FDDI routing and FDDI-to-Ethernet routing, both under minimal system and network load. Note that the data presented for the TCP-based applications shows that throughput is limited by the way TCP calculates its flow control window when data is destined for a remote network. All current TCP implementations have this same limitation because all nonlocal connections must have a small flow control window size of 576 bytes to avoid the saturation of intermediate gateways. Since both Ethernet and FDDI systems can transmit frames larger than this flow control window, the advantage of transmitting maximum-sized frames is lost. UDP does not have this limitation; thus, throughput numbers are only slightly lower than in the nonrouting case.

Future

This section describes some areas of research that may impact the use of the FDDI system. Included are discussions on protocol alternatives, future performance work, and how this system will facilitate new software technologies.

New Protocols

As illustrated in Figure 4, processor speed is the current bottleneck in FDDI throughput. While processor speed continues to increase, emerging protocols are making efficient use of available processing power and are yielding additional gains in network performance.

A development relevant to this discussion of protocol alternatives is the versatile message transport protocol (VMTP) research project from Stanford University.⁴ VMTP demonstrates that a reliable transport is achievable with no greater overhead than existing unreliable transports. VMTP, therefore, represents an alternative to TCP that would nearly double the throughput of some remote procedure call (RPC) applications. Also, knowledge gained from the VMTP research may influence future internet or open systems interconnection (OSI) directions.

Future ULTRIX Network Performance Work

In addition to examining new protocols, performance work is continuing with our existing ULTRIX protocols. One area being studied is data copy. Currently, user data is copied twice as it traverses the internet protocol stack. One copy occurs in the socket subsystem, and the other one takes place in the device driver. Data copies account for 50 percent of CPU utilization time when large amounts of data are transferred. Eliminating one copy can yield increased performance, with savings of at least 25 percent of the total processing time.

An FDDI adapter optimized for the internet protocol stack may also provide improved performance. This decrease in processing time may result from

Table 4 FDDI Routing for a DECstation 5000 Workstation

Transport	Application	Ethernet-to-FDDI Rate (Megabits)	FDDI-to-FDDI Rate (Megabits)
TCP	rcp	4.8	4.8
TCP	FTP	2.7	2.7
UDP/NFS	NFS Read	8.0	16.8
UDP	Spray	9.0	18.0
TCP	TTCP	6.4	6.4

calculating internet checksums in the adapter or from moving the complete protocol stack to the adapter. For example, researchers have proposed protocol engine chips that would off-load all protocol processing to customized chips. With such real-time protocol engines, existing processors could easily outpace current FDDI speeds.

Conclusions

Digital brought FDDI to the ULTRIX workstation to satisfy the growing network demands of its customers. The number of network-intensive applications that run on ULTRIX workstations is growing at a fast pace. Graphics and imaging applications have the potential of generating megabytes of network data. Also, multimedia applications can strain FDDI networks and are not practical using Ethernet. The best scenario for a live motion video application is the requirement of at least a 1-MB, continuous network connection, enough to easily saturate Ethernet. Even a live audio application will require a 200- to 300-KB-per-second network connection. Clearly, with applications that demand these data rates, FDDI bandwidth is necessary.

The DEC FDDIcontroller 700 adapter brings FDDI to the desktop. The adapter is well matched to the DECstation 5000 model 200, joining a 25-mips processor to a 100-Mb-per-second data link. As the next generation of LAN, FDDI extends the base for network applications by allowing those applications that run on Ethernet to run faster and by providing the bandwidth for a whole new generation of applications. FDDI is the network of the '90s, as Ethernet was the network of the '80s.

Acknowledgments

We would like to thank Kathy Wilde for her early participation on the project, Kent Ferson for general support and encouragement, and Fred Glover for his help in reviewing the document. Nolan Ring patiently edited the early draft. Julia Fey helped with the collection of performance data and with the data analysis. We would especially like to thank the staff of the *Digital Technical Journal* for their help and support.

References

1. *A Proposed Standard for the Transmission of IP Datagrams over FDDI Networks*, Internet Network Working Group, RFC 1188 (October 1990).
2. *Fiber Distributed Data Interface (FDDI)-Token Ring Media Access Control (MAC)*, ANSI X3.139-1987.
3. *ANSI FDDI Station Management Standard*, ANSI X3T9.5, Revision 5.1 (September 1989).
4. W. Mason, "VMTP: A High Performance Transport Protocol," *Connexions*, vol. 4, no. 6 (June 1990): 2-10.

General References

A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, Internet Network Working Group, RFC 1042 (February 1988).

ANSI/IEEE Standards for Local Area Networks: Logical Link Control, ANSI/IEEE Standard 802.2-1985 (New York: The Institute of Electrical and Electronics Engineers, Inc., 1985).

Further Readings

The Digital Technical Journal publishes papers that explore the technological foundations of Digital's major products. Each Journal focuses on at least one product area and presents a compilation of papers written by the engineers who developed the product. The content for the Journal is selected by the Journal Advisory Board.

Topics covered in previous issues of the *Digital Technical Journal* are as follows:

Transaction Processing, Databases, and Fault-tolerant Systems

Vol. 3, No. 1, Winter 1991

The architecture and products of Digital's distributed transaction processing systems, with information on monitors, performance measurement, system sizing, database availability, commit processing, and fault tolerance

VAX 9000 Series

Vol. 2, No. 4, Fall 1990

The technologies and processes used to build Digital's first mainframe computer, including papers on the architecture, microarchitecture, chip set, vector processor, and power system, as well as CAD and test methodologies

DECwindows Program

Vol. 2, No. 3, Summer 1990

An overview and descriptions of the enhancements Digital's engineers have made to MIT's X Window System in such areas as the server, toolkit, interface language, and graphics, as well as contributions made to related industry standards

VAX 6000 Model 400 System

Vol. 2, No. 2, Spring 1990

The highly expandable and configurable midrange family of VAX systems that includes a vector processor, a high-performance scalar processor, and advances in chip design and physical technology

Compound Document Architecture

Vol. 2, No. 1, Winter 1990

The CDA family of architectures and services that support the creation, interchange, and processing of compound documents in a heterogeneous network environment

Distributed Systems

Vol. 1, No. 9, June 1989

Products that allow system resource sharing throughout a network, the methods and tools to evaluate product and system performance

Storage Technology

Vol. 1, No. 8, February 1989

Engineering technologies used in the design, manufacture, and maintenance of Digital's storage and information management products

CVAX-based Systems

Vol. 1, No. 7, August 1988

CVAX chip set design and multiprocessing architecture of the midrange VAX 6200 family of systems and the MicroVAX 3500/3600 systems

Software Productivity Tools

Vol. 1, No. 6, February 1988

Tools that assist programmers in the development of high-quality, reliable software

VAXcluster Systems

Vol. 1, No. 5, September 1987

System communication architecture, design and implementation of a distributed lock manager, and performance measurements

VAX 8800 Family

Vol. 1, No. 4, February 1987

The microarchitecture, internal boxes, VAXBI bus, and VMS support for the VAX 8800 high-end multiprocessor, simulation, and CAD methodology

Networking Products

Vol. 1, No. 3, September 1986

The Digital Network Architecture (DNA), network performance, LANbridge 100, DECnet-ULTRIX and DECnet-DOS, monitor design

MicroVAX II System

Vol. 1, No. 2, March 1986

The implementation of the microprocessor and floating point chips, CAD suite, MicroVAX workstation, disk controllers, and TK50 tape drive

VAX 8600 Processor

Vol. 1, No. 1, August 1985

The system design with pipelined architecture, the I-box, F-box, packaging considerations, signal integrity, and design for reliability

Subscriptions to the *Digital Technical Journal* are available on a yearly, prepaid basis. The subscription rate is \$40.00 per year (four issues). Requests

should be sent to Cathy Phillips, Digital Equipment Corporation, MLO1-3/B68, 146 Main Street, Maynard, MA 01754, U.S.A. Subscriptions must be paid in U.S. dollars, and checks should be made payable to Digital Equipment Corporation.

Single copies and past issues of the *Digital Technical Journal* can be ordered from Digital Press at a cost of \$16.00 per copy.

Technical Papers by Digital Authors

E. Atakov, "Electromigration Resistance of TiN-layered Ti Doped Al Interconnects," *IEEE VLSI Multilevel Conference* (June 1990).

R. Bansal, H. Fu, C. Haggerty, and P. Posco, "Hyper-information Systems and Technology Transfer," *ASME Computers in Engineering* (August 1990).

D. Benson, P. Ciarfella, and P. Hayden, "FDIC: Meeting the Interoperability Challenge," *IEEE Local Computer Networks* (October 1990).

D. Bhandarkar and R. Brunner, "VAX Vector Architecture," *International Symposium on Computer Architecture* (May 1990).

G. Champine and D. Geer, "Project ATHENA as a Distributed Computer System," *IEEE Computer* (September 1990).

R. Collica, "The Physical Mechanisms of Defect Clustering and Its Correlation to Yield Model Parameters for Yield Improvement," *Advanced Semiconductor Manufacturing Conference* (September 1990).

R. Csencsits, J. Rose, N. Riel, and J. Dion, "Cross Section Preparation of CU/PI Interfaces for TEM Analyses," *Electron Microscopy '90* (August 1990).

D. Dalal, "A 500 KHz Multi-output Converter with Zero Voltage Switching," *IEEE Applied Power Electronics* (March 1990).

S. Das and W. Chaurette, "Self Heating Effects on MR Head Performance," *IEEE Intermag '90* (April 1990).

B. Doyle and K. Mistry, "The Generation and Characterization of Electron and Hole Traps Created by Hole Injection during Low Gate Voltage Hot-carrier Stressing of N-MOS Transistors," *IEEE Transactions on Electron Devices* (August 1990).

A. Enver and J. Clement, "Finite-element Numerical Modeling of Currents in VLSI Interconnects," *IEEE VLSI Multilevel Conference* (June 1990).

A. Gardel and P. Deosthali, "Using Simulation in Semiconductor Fabrication," *Electro '90* (May 1990).

C. Gilbert, "Improving Distance Learning in Industry," *College Industry Education Conference* (February 1990).

H. Hvostov and M. Luksic, "On the Structure of H-Infinity Controllers," *American Control Conference* (May 1990).

A. Incorvaia, "Case Studies in Software Reuse," *IEEE Software Applications Conference* (November 1990).

K. Mistry and B. Doyle, "The Role of the Electron Trap Created in Enhanced Hot-carrier Degradation during AC Stress," *IEEE Electron Device Letters* (June 1990).

C. Nofsinger and Z. Cvetanovic, "Parallel Astar on Message-passing Architectures," *International Conference on System Sciences* (June 1990).

S. Novotny, "Performance Evaluation of a Gifford-McMahon Refrigerator for Cryogenically Cooled Computer Applications," *I-Therm II* (May 1990).

P. Riley and T. Clark, "Integrated Deposition and Etchback of Tungsten in a Multichamber, Single Wafer System," *IEEE VLSI Multilevel Conference* (June 1990).

M. Sidman, "Parametric System Identification on Logarithmic Frequency Response Data," *American Control Conference* (May 1990).

E. Zimran, "Generic Material Handling System," *International Conference on CIM* (May 1990).

Digital Press

Digital Press is the book publishing group of Digital Equipment Corporation. The Press is an international publisher of computer books and journals on new technologies and products for users, system and network managers, programmers and other professionals. Proposals and ideas for books in these and related areas are welcomed.

VAX/VMS: Writing Real Programs in DCL

Paul C. Anagnostopoulos, 1989, softbound, 409 pages (\$29.95)

X WINDOW SYSTEM TOOLKIT: The Complete Programmer's Guide and Specification

Paul J. Asente and Ralph R. Swick, 1990, softbound, 967 pages (\$44.95)

UNIX FOR VMS USERS

Philip E. Bourne, 1990, softbound,
368 pages (\$28.95)

**VAX ARCHITECTURE REFERENCE MANUAL,
Second Edition**

Richard A. Brunner, Editor, 1991, softbound,
560 pages (\$44.95)

**SOFTWARE DESIGN TECHNIQUES FOR LARGE
ADA SYSTEMS**

William E. Byrne, 1991, hardbound,
314 pages (\$44.95)

**INFORMATION TECHNOLOGY STANDARDIZA-
TION: Theory, Practice, and Organizations**

Carl F. Cargill, 1989, softbound,
252 pages (\$24.95)

**THE DIGITAL GUIDE TO SOFTWARE
DEVELOPMENT**

Corporate User Publication Group of Digital
Equipment Corporation, 1990, softbound,
239 pages (\$27.95)

**DIGITAL GUIDE TO DEVELOPING
INTERNATIONAL SOFTWARE**

Corporate User Publication Group of Digital
Equipment Corporation, 1991, softbound,
400 pages (\$28.95)

**VAX/VMS INTERNALS AND DATA
STRUCTURES: Version 5.2**

Ruth E. Goldenberg and Lawrence J. Kenah, with
the assistance of Denise E. Dumas, 1991, hardbound,
approximately 1,300 pages (\$124.95)

**COMPUTER PROGRAMMING AND
ARCHITECTURE: The VAX, Second Edition**

Henry M. Levy and Richard H. Eckhouse Jr., 1989,
hardbound, 444 pages (\$38.00)

**USING MS-DOS KERMIT: Connecting Your PC
to the Electronic World**

Christine M. Gianone, 1990, softbound,
244 pages, with Kermit Diskette (\$29.95)

**THE USER'S DIRECTORY OF COMPUTER
NETWORKS**

Tracy L. LaQuey, 1990, softbound,
630 pages (\$34.95)

SOLVING BUSINESS PROBLEMS WITH MRP II

Alan D. Luber, 1991, hardbound,
333 pages (\$34.95)

VMS FILE SYSTEM INTERNALS

Kirby McCoy, 1990, softcover,
460 pages (\$49.95)

**TECHNICAL ASPECTS OF DATA
COMMUNICATION, Third Edition**

John E. McNamara, 1988, hardbound,
383 pages (\$42.00)

LISP STYLE and DESIGN

Molly M. Miller and Eric Benson, 1990, softbound,
214 pages (\$26.95)

THE VMS USER'S GUIDE

James F. Peters III and Patrick J. Holmay, 1990,
softbound, 304 pages (\$28.95)

**THE MATRIX: Computer Networks and
Conferencing Systems Worldwide**

John S. Quarterman, 1990, softbound,
719 pages (\$49.95)

X AND MOTIF QUICK REFERENCE GUIDE

Randi J. Rost, 1990, softbound,
369 pages (\$24.95)

**FIFTH GENERATION MANAGEMENT:
Integrating Enterprises Through Human
Networking**

Charles M. Savage, 1990, hardbound,
267 pages (\$28.95)

**A BEGINNER'S GUIDE TO VAX/VMS UTILITIES
AND APPLICATIONS**

Ronald M. Sawey and Troy T. Stokes, 1989,
softbound, 278 pages (\$26.95)

X WINDOW SYSTEM, Second Edition

Robert Scheifler and James Gettys, 1990,
softbound, 851 pages (\$49.95)

COMMON LISP: The Language, Second Edition

Guy L. Steele Jr., 1990, 1,029 pages (\$38.95 in
softbound, \$46.95 in hardbound)

WORKING WITH WPS-PLUS

Charlotte Temple and Dolores Cordeiro, 1990,
softbound, 235 pages (\$24.95)

To receive a copy of our latest catalog or informa-
tion on these or other publications from Digital
Press, write or call:

Digital Press
Department DTJ
12 Crosby Drive
Bedford, MA 01730
617/276-1536

Or, to order, call DECdirect at
800-DIGITAL (800-344-4825).

digital™

ISSN 0898-901X

Printed in U.S.A. EYH876E-DP/91 03 02 16.0 MCG/BUO Copyright © Digital Equipment Corporation. All Rights Reserved.