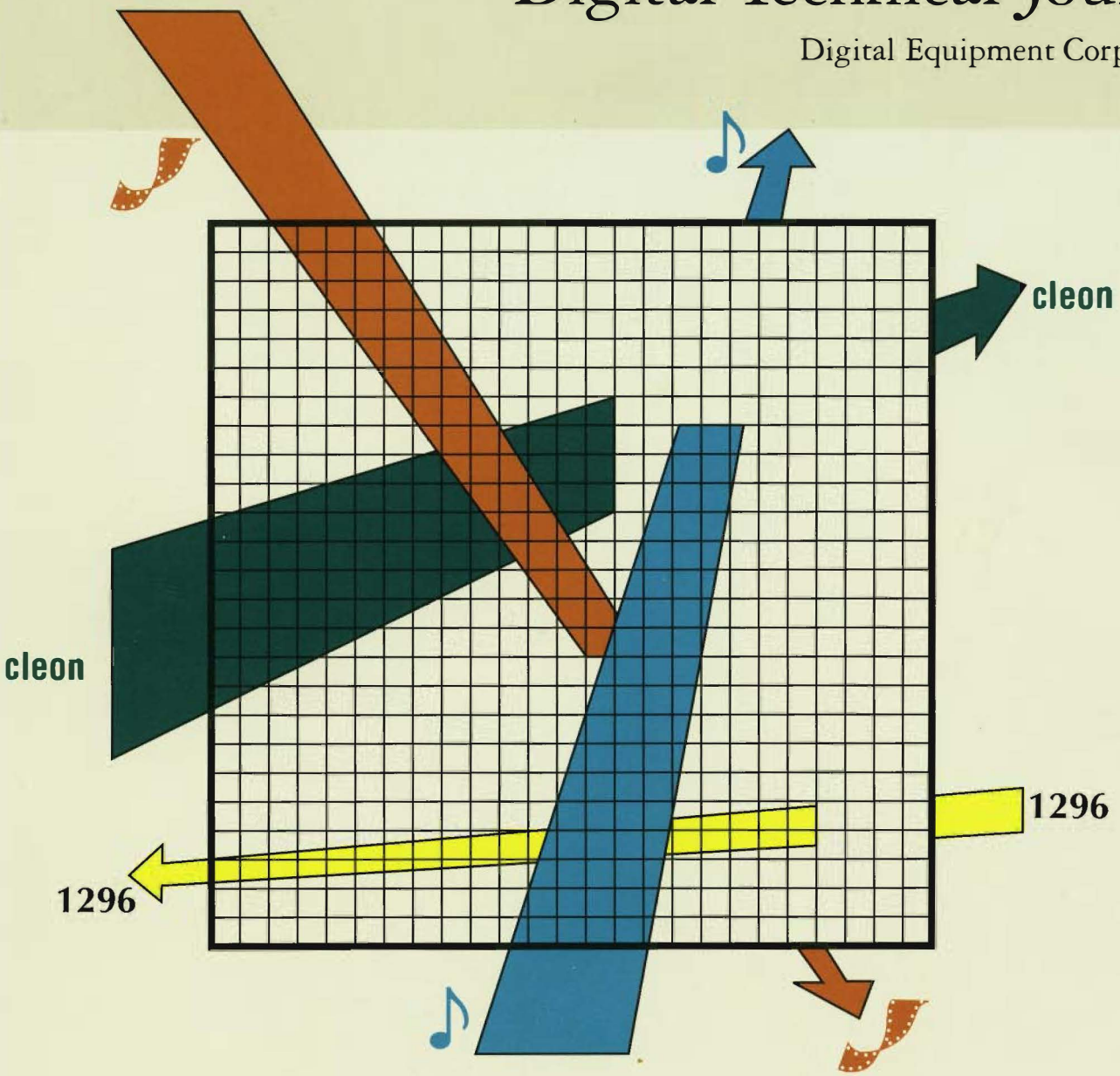
- *High-performance Networking*
- *OpenVMS AXP System Software*
- *Alpha AXP PC Hardware*

Digital Technical Journal

Digital Equipment Corporation



Volume 6 Number 1

Winter 1994

The *Digital Technical Journal* is a refereed journal published quarterly by Digital Equipment Corporation, 30 Porter Road LJO2/D10, Littleton, Massachusetts 01460. Subscriptions to the *Journal* are $40.00 (non-U.S. $60) for four issues and $75.00 (non-U.S. $115) for eight 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 by calling DECdirect at 1-800-DIGITAL (1-800-344-4825). Recent back issues of the *Journal* are also available on the Internet at gatekeeper.dec.com in the directory /pub/DEC/DECinfo/DTJ.

Digital employees may order subscriptions through Readers Choice by entering VTX PROFILE at the system prompt.

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

ISSN 0898-901X

Documentation Number EY-Q011E-TJ

Book production was done by Quantic Communications, Inc.

Cover Design

The GIGAswitch system is a high-performance packet switch for interconnecting a variety of network types and components, including LANs, WANs, clients, servers, and members of workstation farms. The grid on our cover represents the crossbar switching fabric at the core of the GIGAswitch system. Data represented here as flowing through the crossbar —numeric, audio, video, text—is typical of applications that use the GIGAswitch system.

The cover was designed by Joe Pozerycki, Jr., of Digital's Corporate Design Group.

Contents

Editor's Introduction

Jane C. Blake
Managing Editor

In answer to a recent *Journal* survey question, half of those who responded said they would like to see a variety of topics in each issue; half prefer the single-topic format. This issue of the *Digital Technical Journal* will please those who like variety. The papers herein address networking, database performance, software processes, PCs, and chips. In the future, we will try to please readers on both sides of the question by presenting single-topic issues, such as Workflow Software upcoming, and issues that encompass a range of topics.

Our opening paper is about the GIGAswitch packet-switching system, used to increase data transfer among interconnected LANs and in workstation farms. At 6.25 million connections per second, GIGAswitch is among the industry's fastest multiport packet-switching systems. The paper by Bob Souza, P. G. Krishnakumar, Cüneyt Özveren, Bob Simcoe, Barry Spinney, Bob Thomas, and Bob Walsh is a substantive overview of GIGAswitch, the first implementation of which includes an FDDI bridge. The authors discuss the major design issues, including the data link independent crossbar, the arbitration algorithm (called take-a-ticket), and the techniques used to ensure the robustness, reliability, and cost-effectiveness of the switch.

Just as critical as network performance in high-end system environments, such as stock exchanges and banking, is the performance of server-based database management systems. In their paper on the DEC Rdb version 6.0 database, Lou Dimino, Rabah Mediouni, T. K. Rengarajan, Mike Rubino, and Peter Spiro examine earlier steps taken to establish good database performance on AXP systems and compare these and traditional approaches with the new enhancements that shorten the code paths, minimize I/O operations, and reduce stall times. Notable among performance results was a world record for transactions per second in a test of the Rdb version 6.0 database running on a DEC 10000 AXP system.

Data availability is another important characteristic of high-end systems. Volume shadowing is a feature of the OpenVMS operating system that increases availability by replicating data on disk storage devices. The port of Volume Shadowing Phase II to OpenVMS AXP is a case study of how a small engineering team can use software development processes to tackle complex system design tasks and deliver improved quality products on accelerated first-release schedules. Bill Goleman, Robert Thomson, and Paul Houlihan describe key process innovations, including early code inspections and profile testing, which simulates complex scenarios in order to reveal errors.

In the first of two papers on Alpha AXP PC products, Dave Conroy, Tom Kopec, and Joe Falcone trace the design choices, alternatives, and methodology used in the evolutionary development of the first AXP PCs. The authors explore the lessons learned from designing two experimental systems; the first demonstrated the feasibility of building a PC utilizing the DECchip 21064 microprocessor and industry-standard components; the subsequent system incorporated the EISA bus. They then review the design of the DECpc AXP 150 product, which built on the successes of the experimental systems and was completed in just 12 weeks.

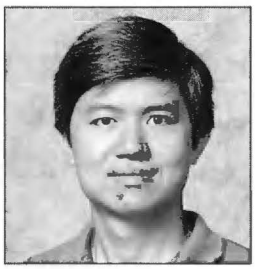
Future Alpha AXP PCs and desktop systems will use new Alpha AXP microprocessors that have higher levels of system integration for performance yet employ packaging and clocking techniques that reduce system cost. Dina McKinney, Masooma Bhaiwala, Kwong Chui, Chris Houghton, Jim Mullens, Dan Leibholz, Sanjay Patel, Del Ramey, and Mark Rosenbluth explain the trade-offs and results of the 21066 design, which is the first microprocessor to integrate a PCI bus controller along with many other system-level functions.

At the close of this issue, we have listed the referees who offered their expert advice on the content and readability of papers submitted to the *Journal* between January 1993 and February 1994. We are grateful for the specialized knowledge they have shared with authors and editors alike.

Jane Blake

Biographies

Masooma Bhaiwala A hardware engineer in the Models, Tools, and Verification Group of Semiconductor Engineering, Masooma Bhaiwala has contributed to several projects since joining Digital in 1991. She worked on the I/O controller verification of the DECchip 21066 microprocessor and wrote the PCI transactor that mimics Intel's PCI bus protocol. Recently, she led the DECchip 21066 second-pass verification efforts. Masooma received a B.S.C.E. from N.E.D. University of Engineering and Technology, Pakistan, and an M.S.C.E. from the University of Massachusetts at Lowell.

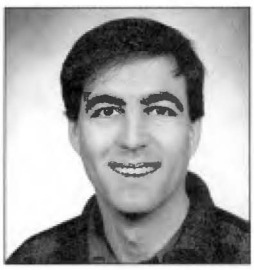


Kwong-Tak A. Chui Kwong Chui, a principal engineer in the Semiconductor Engineering Group, is currently leading the definition of a core chip set for a RISC processor. He was the co-architect of the PCI interface section of the DECchip 21066. Since joining Digital in 1985, Kwong has worked on memory and I/O bus controller chips for the MicroVAX 3000 and VAX 4000 series systems, and a cache address fanout buffer and multiplexer chip for the DEC 4000 AXP system. Kwong holds a B.S. in computer engineering from the University of Illinois at Urbana-Champaign and an M.Eng.E.E. from Cornell University.

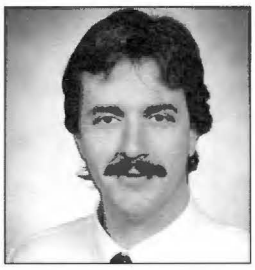
David G. Conroy Dave Conroy is a senior consulting engineer at Digital's System Research Center in Palo Alto, California. He received a B.A.Sc. degree in electrical engineering from the University of Waterloo, Canada, and moved to the United States in 1980 as a cofounder of the Mark Williams Company. He joined Digital in 1983 to work on the DECtalk speech synthesis system and became a member of the Semiconductor Engineering Group in 1987 to work on system-level aspects of RISC microprocessors, including the DECchip 21064 CPU. Dave is now investigating the design of high-performance memory systems.

Lucien A. Dimino A principal software engineer, Lucien Dimino is a member of the Database Systems Group. He is responsible for the RMU relational database management utility. Formerly with AT&T Bell Telephone Laboratories, he joined Digital in 1974. At Digital he has worked in the areas of networking and communications, transaction processing, manufacturing productivity and automation, and database management. Lucien received a B.S. in mathematics (1966) from City College of New York and an M.S. in mathematics from Stevens Institute of Technology.

Joseph R. Falcone Joseph Falcone is a hardware engineering manager in the Technical Original Equipment Manufacturer Group, which develops Alpha AXP board products for the OEM market. Prior to this, he developed the system architecture and business strategy for the DECpc AXP 150 personal computer. Joe came to Digital from Hewlett-Packard Laboratories in 1983 to work in Corporate Research & Architecture on distributed systems research and development. He received an A.B.C.S. (1980) from the University of California, Berkeley, and an M.S.E.E. (1983) from Stanford University.



William L. Goleman Principal software engineer Bill Goleman led the project to port the Volume Shadowing Phase II product to OpenVMS AXP. He is currently involved in planning OpenVMS I/O subsystem strategies. Bill joined Digital's Software Services organization in 1981. He worked in the Ohio Valley District and then moved to VMS Engineering in 1985. Since then Bill has worked on the MSCP server, local area VAXcluster, mixed-architecture clusters, volume shadowing, and the port of cluster I/O components to OpenVMS AXP. Bill received a B.S. in computer science from Ohio State University in 1979.



Christopher L. Houghton A principal hardware engineer in the Semiconductor Engineering Group, Chris Houghton is responsible for the signal integrity, packaging, and I/O circuit design of the DECchip 21066 Alpha AXP microprocessor. His previous work includes signal integrity, packaging, and circuit design for the GIGAswitch crossbar chips, digital and analog circuit design for a semicustom cell library, and advanced development of processor-memory interconnect chips. Chris came to Digital in 1985 after receiving a B.S.E.E. from the University of Vermont. He holds one patent.

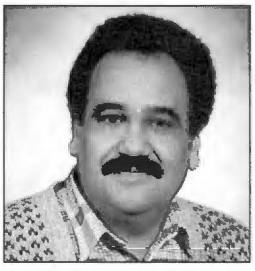
Paul J. Houlihan As a principal engineer with OpenVMS AXP Engineering, Paul Houlihan participated in the port of OpenVMS software components from the VAX to the Alpha AXP architecture, including volume shadowing, the tape class driver, and the system communication services layer. Paul's recent focus has been on the OpenVMS I/O subsystem and software quality. He is the creator of Faulty Towers, a test tool that injects software faults into VMScluster systems. Paul received a B.A. in computer science and a B.A. in political science from the University of Wisconsin at Madison.

Thomas E. Kopec Principal engineer Tom Kopec is a member of the Assistive Technology Group working on compact speech-synthesis and text-to-speech systems. Previously, he was a member of the Entry Systems Business Group that designed the VAX 4000 Models 200 through 600 and the MicroVAX 3500 and 3800 systems. Tom received a B.S.E.C.E. (1980, with honors) in microwave engineering and signal processing and an M.S.E.C.E. (1985) in image processing and computer graphics, both from the University of Massachusetts at Amherst.

P. G. Krishnakumar A principal engineer in Digital's High Performance Networks Group, P. G. Krishnakumar is currently the project leader for the Ethernet-ATM bridge firmware. Prior to this, he worked on the GIGAswitch firmware and performance analysis of the CI and storage systems. He has also worked in the Tape and Optical Systems and the High Availability Systems Groups. He received a B.Tech. in electrical engineering from the Institute of Technology-BHU, India, an M.S. in operation research and statistics, and an M.S. in computer engineering, both from Rensselaer Polytechnic Institute.

Daniel L. Leibholz Daniel Leibholz, a senior engineer in the Semiconductor Engineering Group, has been designing integrated Alpha AXP CPUs at Digital for three years. He is an architect of the DECchip 21066 Alpha AXP microprocessor and is currently involved in the design of a high-performance microprocessor. While at Digital he has also worked on the development of massively parallel processors and was awarded a patent for a processor allocation algorithm. Daniel joined Digital in 1988 after earning B.S. and M.S. degrees in electrical engineering from Brown University in Providence, Rhode Island.

Dina L. McKinney Principal engineer Dina McKinney supervises the DECchip 21066 low-cost Alpha AXP verification team. Since joining Digital in 1981, Dina has held design and verification positions on various projects, including PC modules, semicustom chips, and full-custom chips for communication, graphics, and processor products. Previously, Dina was a member of the U.S. Air Force. She holds two technical degrees from Community College of the Air Force and Gulf Coast College, a B.S.E.E. (1982, summa cum laude) from Central New England College, and an M.S.E.E. (1990) from Worcester Polytechnic Institute.



Rabah Mediouni Rabah Mediouni joined Digital in 1980 and is currently a principal engineer in the Rdb Consulting Group. His responsibilities include consulting on database design and tuning and performance characterization of new features in major Rdb releases. He was responsible for the performance characterization effort during the DEC Rdb port to the AXP platforms. Prior to this, Rabah worked in the Mid-range Systems Performance Group as a primary contributor to the VAX 8000 series performance characterization project. Rabah received an M.S. in computer science from Rivier College in 1985.

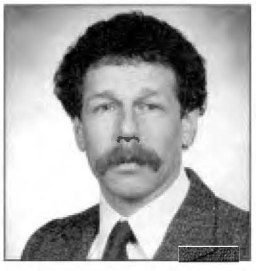
James R. Mullens James Mullens is a principal hardware engineer in the Semiconductor Engineering Group. He is working on CPU advanced development and is responsible for the architecture and design of the DECchip 21066 dynamic memory controller. In earlier work, he performed verification and test of the system-on-a-chip (SOC), designed four ASIC devices used in workstations and DECwindows terminals, led the Corporate Mouse and Tablet projects, and was a design and project engineer for the Mini-Exchange. Jim joined Digital in 1982 after receiving a B.S.E.E. from Northeastern University.

Cüneyt M. Özveren Cüneyt Özveren joined Digital in 1990 and is a principal engineer in the Networks Engineering Advanced Development Group. He is currently working on all-optical networks and ATM architecture. His previous work included the architecture, design, and implementation of the SCP hardware in GIGAswitch. He received a Ph.D. (1989) in electrical engineering and computer science from MIT and an M.S. (1989) from MIT's Sloan School of Management. His research included the analysis and control of large-scale dynamic systems, with applications to communication systems, manufacturing, and economics.

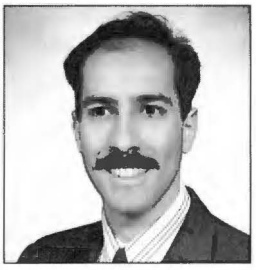
Sanjay J. Patel As a verification engineer for the DECchip 21066 low-cost Alpha AXP project, Sanjay Patel was a key implementor of the DECchip 21066 verification strategy. He is currently working on specifying power management capabilities for a future chip. Sanjay joined Digital in 1992. In his earlier experience as a co-op student at IBM in Kingston, New York, he helped develop a vectorized subroutine library. Sanjay received B.S.E. (1990) and M.S.E. (1992) degrees in computer engineering from the University of Michigan.

Delvan A. Ramey A principal hardware engineer, Del Ramey specializes in analog CMOS development for CPU, communications, and graphics chips. His recent work involves phase-locked loop (PLL) design. Del architected the hybrid analog-to-digital PLL clock and data recovery system in the Ethernet protocol DECchip 21040 product, and designed the frequency synthesis PLL for the DECchip 21066 Alpha AXP microprocessor. He has also developed semicustom chips and libraries. Del joined Digital in 1980, after completing his Ph.D. at the University of Cincinnati. He is a member of IEEE, Eta Kappa Knu, and Phi Kappa Phi.

T. K. Rengarajan T. K. Rengarajan, a member of the Database Systems Group since 1987, works on the KODA software kernel of the DEC Rdb system. He has contributed in the areas of buffer management, high availability, OLTP performance on Alpha AXP systems, and multimedia databases. He designed high-performance logging, recoverable latches, asynchronous batch writes, and asynchronous prefetch features for DEC Rdb version 6.0. Ranga holds M.S. degrees in computer-aided design and computer science from the University of Kentucky and the University of Wisconsin, respectively.



Mark B. Rosenbluth Consulting engineer Mark Rosenbluth contributes to the architecture, modeling, and logic design of the DECchip 21066 microprocessor. In earlier work, he was involved in the modeling, logic design, and circuit design of the DC222 system-on-a-chip (SOC) VAX microprocessor, the DC358 MicroVAX DMA controller chip, and the J-11 PDP-11 microprocessor. He also helped create semicustom design system and cell libraries. Mark came to Digital in 1977 from RCA Corporation. He holds a B.S.E.E. from Rutgers University College of Engineering.



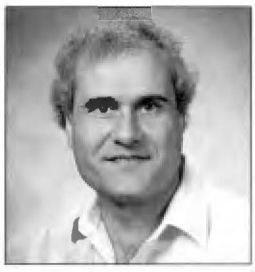
Michael S. Rubino Michael Rubino joined Digital in 1983 and is a principal software engineer in the Database Systems Group. As the Alpha program manager for database systems, his primary role is the oversight of the porting of Rdb as well as other database products from VAX to Alpha AXP. Prior to this work, Michael was with VMS Engineering and contributed to the RMS and RMS/journaling projects. Prior to that, he was the KODA (database kernel) project leader. Michael received a B.S. in computer science from the State University of New York at Stony Brook in 1983.

Robert J. Simcoe Robert Simcoe is a consulting engineer in the High Performance Networks Group and is responsible for ATM switch products. Prior to this, he worked in Network Engineering Advanced Development on the GIGAswitch and SONET/ATM. He has also worked on chip designs, from MicroVAX and Scorpio FPUs to high-speed switching for network switches. Prior to joining Digital in 1983, he designed ICs and small systems for General Electric; before that, he designed secure communications devices for the National Security Agency. He has published several papers and holds 14 patents.

Robert J. Souza Bob Souza is a consulting engineer in the Networks Engineering Advanced Development Group. He was the project leader for the GIGAswitch SCP module. Since joining Digital in 1982, Bob has worked on the DEMPR Ethernet repeater for MicroSystems Advanced Development and has coimplemented an RPC system for Corporate Research at Project Athena. He is the author of several tools for sequential network design, a number of papers, and several patents. Bob received a Ph.D. in computer science from the University of Connecticut.

Barry A. Spinney Barry Spinney is a consulting engineer in the Network Engineering Advanced Development Group. Since joining Digital in 1985, Barry has been involved in the specification and/or design of FDDI chip sets (MAC, RMC, and FCAM chips) and the GIGAswitch chips (FGC, GCAM, GPI, and CBS chips). He also contributed to the GIGAswitch system architecture and led the GIGAswitch FDDI line card design (chips, hardware, and firmware). He holds a B.S. in mathematics and computer science from the University of Waterloo and an M.S.C.S. from the University of Toronto. Barry is a member of IEEE.

Peter M. Spiro Peter Spiro, a consulting software engineer, is currently the technical director for the Rdb and DBMS software product set. Peter's current focus is very large database issues as they relate to the information highway. Peter joined Digital in 1985, after receiving M.S. degrees in forest science and computer science from the University of Wisconsin-Madison. He has five patents related to database journaling and recovery, and he has authored three papers for earlier issues of the *Digital Technical Journal*. In his spare time, Peter is building a birch bark canoe.



Robert E. Thomas Bob Thomas received a Ph.D. in computer science from the University of California, Irvine. He has worked on fine-grained dataflow multiprocessor architectures at the MIT Lab for Computer Science. After joining Digital in 1983, Bob pioneered the development of a cell-based network switching architecture and deadlock-free routing. He is a coinventor of the high-speed crossbar arbitration method for GIGAswitch and was co-project leader of the AToM-3 gate array and the ATM/SONET/T3/E3 interface card. Bob is currently hardware project leader for a 622-Mbs ATM PCI adapter.

Robert G. Thomson A senior software engineer in the OpenVMS AXP Group, Robert Thomson led the validation of volume shadowing's port to the Alpha AXP platform. During the OpenVMS port, he measured quality and contributed to functional verification, for which he was co-recipient of an Alpha AXP Achievement Award. Since joining Digital in 1986, Robert has also contributed to improvements in system availability measurement and in symmetric multiprocessing and backup performance. He has a patent and three published papers based on this work. Robert holds an M.S. in computer engineering from Boston University.

Robert J. Walsh Bob Walsh has published papers on TCP/IP networking, on radiosity and ray-tracing algorithms for 3-D shaded graphics, and on graphics rendering with parallel processors. He has a patent pending on language constructs and translation for programming parallel machines. Bob has extensive operating system experience for uniprocessors (UNIX V.6, 4.1BSD, and follow-ons) and switch-based parallel processors (BBN's Butterfly and Digital's GIGAswitch). He received A.B., S.M., and Ph.D. degrees from Harvard University.

Robert J. Souza
P. G. Krishnakumar
Cüneyt M. Özveren
Robert J. Simcoe
Barry A. Spinney
Robert E. Thomas
Robert J. Walsh

GIGAswitch System: A High-performance Packet-switching Platform

The GIGAswitch system is a high-performance packet-switching platform built on a 36-port 100 Mb/s crossbar switching fabric. The crossbar is data link independent and is capable of making 6.25 million connections per second. Digital's first GIGAswitch system product uses 2-port FDDI line cards to construct a 22-port IEEE 802.1d FDDI bridge. The FDDI bridge implements distributed forwarding in hardware to yield forwarding rates in excess of 200,000 packets per second per port. The GIGAswitch system is highly available and provides robust operation in the presence of overload.

The GIGAswitch system is a multiport packet-switching platform that combines distributed forwarding hardware and crossbar switching to attain very high network performance. When a packet is received, the receiving line card decides where to forward the packet autonomously. The ports on a GIGAswitch system are fully interconnected with a custom-designed, very large-scale integration (VLSI) crossbar that permits up to 36 simultaneous conversations. Data flows through 100 megabits per second (Mb/s) point-to-point connections, rather than through any shared media. Movement of unicast packets through the GIGAswitch system is accomplished completely by hardware.

The GIGAswitch system can be used to eliminate network hierarchy and concomitant delay. It can aggregate traffic from local area networks (LANs) and be used to construct workstation farms. The use of LAN and wide area network (WAN) line cards makes the GIGAswitch system suitable for building, campus, and metropolitan interconnects. The GIGAswitch system provides robustness and availability features useful in high-availability applications like financial networks and enterprise backbones.

In this paper, we present an overview of the switch architecture and discuss the principles influencing its design. We then describe the implementation of an FDDI bridge on the GIGAswitch system platform and conclude with the results of performance measurements made during system test.

GIGAswitch System Architecture

The GIGAswitch system implements Digital's architecture for switched packet networks. The architecture allows fast, simple forwarding by mapping 48-bit addresses to a short address when a packet enters the switch, and then forwarding packets based on the short address. A header containing the short address, the time the packet was received, where it entered the switch, and other information is prepended to a packet when it enters the switch. When a packet leaves the switch, the header is removed, leaving the original packet. The architecture also defines forwarding across multiple GIGAswitch systems and specifies an algorithm for rapidly and efficiently arbitrating for crossbar output ports. This arbitration algorithm is implemented in the VLSI, custom-designed GIGAswitch port interface (GPI) chip.

Hardware Overview

Digital's first product to use the GIGAswitch platform is a modular IEEE 802.1d fiber distributed data interface (FDDI) bridge with up to 22 ports.[1] The product consists of four module types: the FDDI line card (FGL), the switch control processor (SCP),

the clock card, and the crossbar interconnection. The modules plug into a backplane in a 19-inch, rack-mountable cabinet, which is shown in Figure 1. The power and cooling systems provide N+1 redundancy, with provision for battery operation.

The first line card implemented for the GIGAswitch system is a two-port FDDI line card (FGL-2). A four-port version (FGL-4) is currently under design, as is a multifunction asynchronous transfer mode (ATM) line card. FGL-2 provides connection to a number of different FDDI physical media using media-specific daughter cards. Each port has a lookup table for network addresses and associated hardware lookup engine and queue manager. The SCP provides a number of centralized functions, including

- Implementation of protocols (Internet protocol [IP], simple network management protocol [SNMP], and IEEE 802.1d spanning tree) above the media access control (MAC) layer
- Learning addresses in cooperation with the line cards

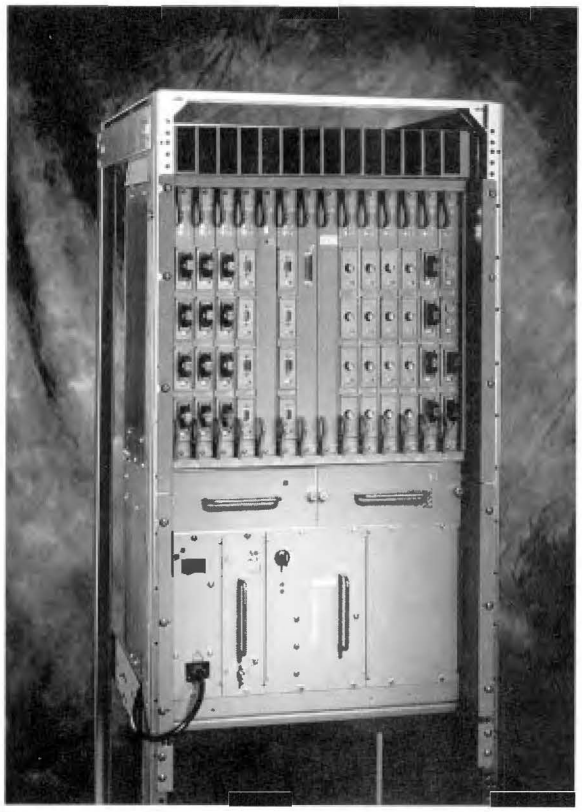


Figure 1 The GIGAswitch System

- Maintaining loosely consistent line card address databases
- Forwarding multicast packets and packets to unknown destinations
- Switch configuration
- Network management through both the SNMP and the GIGAswitch system out-of-band management port

The clock card provides system clocking and storage for management parameters, and the crossbar switch module contains the crossbar proper. The power system controller in the power subsystem monitors the power supply front-end units, fans, and cabinet temperature.

Design Issues

Building a large high-performance system requires a seemingly endless series of design decisions and trade-offs. In this section, we discuss some of the major issues in the design and implementation of the GIGAswitch system.

Multicasting

Although very high packet-forwarding rates for unicast packets are required to prevent network bottlenecks, considerably lower rates achieve the same result for multicast packets in extended LANs. Processing multicast packets on a host is often done in software. Since a high rate of multicast traffic on a LAN can render the connected hosts useless, network managers usually restrict the extent of multicast packets in a LAN with filters. Measuring extended LAN backbones yields little multicast traffic.

The GIGAswitch system forwards unicast traffic in a distributed fashion. Its multicast forwarding implementation, however, is centralized, and software forwards most of the multicast traffic. The GIGAswitch system can also limit the rate of multicast traffic emitted by the switch. The reduced rate of traffic prevents lower-speed LANs attached to the switch through bridges from being rendered inoperable by high multicast rates.

Badly behaved algorithms using multicast protocols can render an extended LAN useless. Therefore, the GIGAswitch system allocates internal resources so that forward progress can be made in a LAN with badly behaved traffic.

Switch Fabric

The core of the GIGAswitch system is a 100 Mb/s full-duplex crossbar with 36 input ports and 36 output ports, each with a 6-bit data path (36 × 36 × 6). The crossbar is formed from three 36 × 36 × 2 custom VLSI crossbar chips. Each crossbar input is paired with a corresponding output to form a dual-simplex data path. The GIGAswitch system line cards and SCP are fully interconnected through the crossbar. Data between modules and the crossbar can flow in both directions simultaneously.

Using a crossbar as the switch connection (rather than, say, a high-speed bus) allows cut-through forwarding: a packet can be sent through the crossbar as soon as enough of it has been received to make a forwarding decision. The crossbar allows an input port to be connected to multiple output ports simultaneously; this property is used to implement multicast. The 6-bit data path through the crossbar provides a raw data-path speed of 150 Mb/s using a 25 megahertz (MHz) clock. (Five bits are used to encode each 4-bit symbol; an additional bit provides parity.)

Each crossbar chip has about 87,000 gates and is implemented using complementary metal-oxide semiconductor (CMOS) technology. The crossbar was designed to complement the FDDI data rate; higher data rates can be accommodated through the use of hunt groups, which are explained later in this section. The maximum connection rate for the crossbar depends on the switching overhead, i.e., the efficiency of the crossbar output port arbitration and the connection setup and tear-down mechanisms.

Crossbar ports in the GIGAswitch system have both physical and logical addresses. Physical port addresses derive from the backplane wiring and are a function of the backplane slot in which a card resides. Logical port addresses are assigned by the SCP, which constructs a logical-to-physical address mapping when a line card is initialized. Some of the logical port number space is reserved; logical port 0, for example, is always associated with the current SCP.

Arbitration Algorithm With the exception of some maintenance functions, crossbar output port arbitration uses logical addresses. The arbitration mechanism, called take-a-ticket, is similar to the system used in delicatessens. A line card that has a packet to send to a particular output port obtains a ticket from that port indicating its position in line. By observing the service of those before it, the line card can determine when its turn has arrived and instruct the crossbar to make a connection to the output port.

The distributed arbitration algorithm is implemented by GPI chips on the line cards and SCP. The GPI is a custom-designed CMOS VLSI chip with approximately 85,000 transistors. Ticket and connection information are communicated among GPIs over a bus in the switch backplane. Although it is necessary to use backplane bus cycles for crossbar connection setup, an explicit connection tear down is not performed. This reduces the connection setup overhead and doubles the connection rate. As a result, the GIGAswitch system is capable of making 6.25 million connections per second.

Hunt Groups The GPI allows the same logical address to be assigned to many physical ports, which together form a hunt group. To a sender, a hunt group appears to be a single high-bandwidth port. There are no restrictions on the size and membership of a hunt group; the members of a hunt group can be distributed across different line cards in the switch. When sending to a hunt group, the take-a-ticket arbitration mechanism dynamically distributes traffic across the physical ports comprising the group, and connection is made to the first free port. No extra time is required to perform this arbitration and traffic distribution. A chain of packets traversing a hunt group may arrive out of order. Since some protocols are intolerant of out-of-order delivery, the arbitration mechanism has provisions to force all packets of a particular protocol type to take a single path through the hunt group.

Hunt groups are similar to the channel groups described by Pattavina, but without restrictions on group membership.[2] Hunt groups in the GIGAswitch system also differ from channel groups in that their use introduces no additional switching overhead. Hardware support for hunt groups is included in the first version of the GIGAswitch system; software for hunt groups is in development at this writing.

Address Lookup

A properly operating bridge must be able to receive every packet on every port, look up several fields in the packet, and decide whether to forward or filter (drop) that packet. The worst-case packet arrival rate on FDDI is over 440,000 packets per second per port. Since three fields are looked up per packet, the FDDI line card needs to perform approximately

1.3 million lookups per second per port; 880,000 of these are for 48-bit quantities. The 48-bit lookups must be done in a table containing 16K entries in order to accommodate large LANs. The lookup function is replicated per port, so the requisite performance must be obtained in a manner that minimizes cost and board area. The approach used to look up the fields in the received packet depends upon the number of values in the field.

Content addressable memory (CAM) technology currently provides approximately 1K entries per CAM chip. This makes them impractical for implementing the 16K address lookup table but suitable for the smaller protocol field lookup. Earlier Digital bridge products use a hardware binary search engine to look up 48-bit addresses. Binary search requires on average 13 reads for a 16K address set; fast, expensive random access memory (RAM) would be needed for the lookup tables to minimize the forwarding latency.

To meet our lookup performance goals at reasonable cost, the FDDI-to-GIGAswitch network controller (FGC) chip on the line cards implements a highly optimized hash algorithm to look up the destination and source address fields. This lookup makes at most four reads from the off-chip static RAM chips that are also used for packet buffering.

The hash function treats each 48-bit address as a 47-degree polynomial in the Galois field of order 2, GF(2).[3] The hashed address is obtained by the equation:

$$M(X) \times A(X) \bmod G(X)$$

where $G(X)$ is the irreducible polynomial, $X^{48} + X^{36} + X^{25} + X^{10} + 1$; $M(X)$ is a nonzero, 47-degree programmable hash multiplier with coefficients in GF(2); and $A(X)$ is the address expressed as a 47-degree polynomial with coefficients in GF(2).

The bottom 16 bits of the hashed address is then used as an index into a 64K-entry hash table. Each hash table entry can be empty or can hold a pointer to another table plus a size between 1 to 7, indicating the number of addresses that collide in this hash table entry (i.e., addresses whose bottom 16 bits of their hash are equal). In the case of a size of 1, either the pointer points to the lookup record associated with this address, or the address is not in the tables but happens to collide with a known address. To determine which is true, the remaining upper 32 bits of the hashed address is compared to the previously computed upper 32 bits of the hash of the known address stored in the lookup record. One of the properties of this hash function is that it is a one-to-one and onto mapping from the set of 48-bit values to the same set. As long as the lookup table records are not shared by different hash buckets, comparing the upper 32 bits is sufficient and leaves an additional 16 bits of information to be associated with this known address.

In the case where $1 < size \leq 7$, the pointer stored in the hash bucket points to the first entry in a balanced binary tree of depth 1, 2, or 3. This binary tree is an array sorted by the upper 32 hash remainder bits. No more than three memory reads are required to find the lookup record associated with this address, or to determine that the address is not in the database.

When more than seven addresses collide in the same hash bucket—a very rare occurrence—the overflow addresses are stored in the GIGAswitch content-addressable memory (GCAM). If several dozen overflow addresses are added to the GCAM, the system determines that it has a poor choice of hash multipliers. It then initiates a re-hashing operation, whereby the SCP module selects a better 48-bit hash multiplier and distributes it to the FGLs. The FGLs then rebuild their hash table and lookup tables using this new hash multiplier value. The new hash multiplier is stored in nonvolatile memory.

Packet Buffering

The FDDI line card provides both input and output packet buffering for each FDDI port. Output buffering stores packets when the outgoing FDDI link is busy. Input buffering stores packets during switch arbitration for the desired destination port. Both input and output buffers are divided into separate first-in, first-out (FIFO) queues for different traffic types.

Switches that have a single FIFO queue per input port are subject to the phenomenon known as head-of-line blocking. Head-of-line blocking occurs when the packet at the front of the queue is destined for a port that is busy, and packets deeper in the queue are destined for ports that are not busy.

The effect of head-of-line blocking for fixed-size packets that have uniformly distributed output port destinations can be closely estimated by a simple probability model based on independent trials. This model gives a maximum achievable mean utilization, $U \approx 1 - 1/e = 63.2$ percent, for switches with more than 20 duplex ports. Utilization increases for smaller switches (or for smaller active parts of larger switches) and is approximately

75 percent for 2 active ports. The independent trial assumption has been removed, and the actual mean utilization has been computed.[4] It is approximately 60 percent for large numbers of active ports.

Hunt groups also affect utilization. The benefits of hunt groups on head-of-line blocking can be seen by extending the simple independent-trial analysis. The estimated mean utilization is

$$U_{n,g} = 1 - \sum_{k=0}^{g} \frac{g-k}{g} \binom{ng}{k} \left(\frac{1}{n}\right)^k \left(\frac{n-1}{n}\right)^{ng-k}$$

where n is the number of groups, and g is the hunt group size. In other words, all groups are the same size in this model, and the total number of switch ports is ($n \times g$). This result is plotted in Figure 2 along with simulation results that remove the independent trial assumption. The simulation results agree with the analysis above for the case of only one link in each hunt group. Note that adding a link to a hunt group increases the efficiency of each member of the group in addition to adding bandwidth. These analytical and simulation results, documented in January 1988, also agree with the simulation results reported by Pattavina.[2]

The most important factor in head-of-line blocking is the distribution of traffic within the switch. When all traffic is concentrated to a single output, there is zero head-of-line blocking because traffic behind the head of the line cannot move any more easily than the head of the line can move. To study this effect, we extended the simple independent-trial model. We estimated the utilization when the traffic from a larger set of inputs (for example, a larger set of workstations) is uniformly distributed to a smaller set of outputs (for example, a smaller set of file servers). The result is

$$U_{n,c} = 1 - \left(\frac{n-1}{n}\right)^{cn} \qquad \lim_{n \to \infty} \quad U = 1 - \frac{1}{e^c}$$

where c is the mean concentration factor of input ports to output ports, and n is the number of outputs. This yields a utilization of 86 percent when an average of two inputs send to each output, and a utilization of 95 percent when three inputs send to each output. Note that utilization increases further for smaller numbers of active ports or if hunt groups are used.

Other important factors in head-of-line blocking are the nature of the links and the traffic distribution on the links. Standard FDDI is a simplex link. Simulation studies of a GIGAswitch system model were conducted to determine the mean utilization of a set of standard FDDI links. They have shown that utilization reaches 100 percent, despite head-of-line blocking, when approximately 50 Mb/s of fixed-size packets traffic, uniformly distributed to all FDDI links in the set, is sent into the switch from each FDDI. The reason is that almost 50 percent of the FDDI bandwidth is needed to sink data from the switch; hence the switch data path is only at 50 percent of capacity when the FDDI links are 100 percent utilized. This result also applies to duplex T3 (45 Mb/s) and all slower links. In these situations, the switch operates at well below capacity, with little internal queuing.

A number of techniques can be used to reduce the effect of head-of-line blocking on link efficiency. These include increasing the speed of the switching fabric and using more complicated queuing mechanisms such as per-port output queues or adding lookahead to the queue service. All these techniques raise the cost and complexity of the switch; some of them can actually reduce performance for normal traffic. Since our studies led us to believe that head-of-line blocking occurs rarely in a GIGAswitch system, and if it does, hunt groups are an effective means for reducing head-of-line blocking, we chose not to implement more costly and complex solutions.

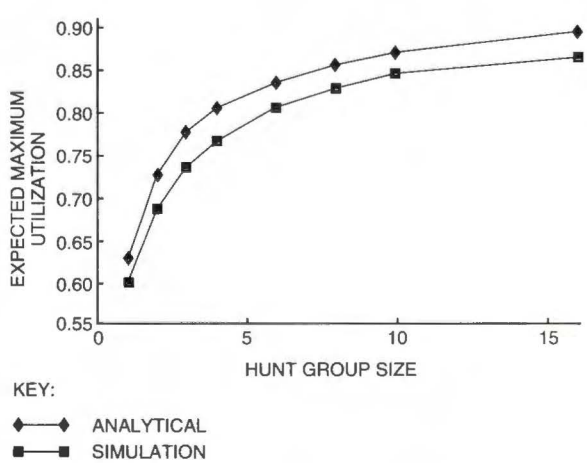


Figure 2 Effect of Hunt Groups on Utilization

Robustness under Overload

The network must remain stable even when the GIGAswitch system is severely stressed. Stability requires timely participation in the 802.1d spanning tree when the packet forwarding loads approach

the worst-case maximum. The techniques used to guarantee forward progress on activities like the spanning tree include preallocation of memory to databases and packets, queuing methods, operating system design, and scheduling techniques. Solutions that provide only robustness are insufficient; they must also preserve high throughput in the region of overload.

Switch Control Processor Queuing and Quota Strategies The SCP is the focal point for many packets, including (1) packets to be flooded, (2) 802.1d control packets, (3) intrabox intercard command (ICC) packets, and (4) SNMP packets.[5] Some of these packets must be processed in a timely manner. The 802.1d control packets are part of the 802.1d algorithms and maintain a stable network topology. The ICCs ensure correct forwarding and filtering of packets by collecting and distributing information to the various line cards. The SNMP packets provide monitoring and control of the GIGAswitch system.

Important packets must be distinguished and processed even when the GIGAswitch system is heavily loaded. The aggregate forwarding rate for a GIGAswitch system fully populated with FGL-2 line cards is about 4 million packets per second. This is too great a load for the SCP CPU to handle on its own. The FDDI line cards place important packets in a separate queue for expedient processing. Special hardware on the SCP is used to avoid loss of important packets.

The crossbar access control (XAC) hardware on the SCP is designed to avoid the loss of any important packet under overload. To distinguish the packets, the XAC parses each incoming packet. By preallocating buffer memory to each packet type, and by having the hardware and software cooperate to maintain a strict accounting of the buffers used by each packet type, the SCP can guarantee reception of each packet type.

Arriving packets allocated to an exhausted buffer quota are dropped by the XAC. For instance, packets to be flooded arrive due to external events and are not rate limited before they reach the SCP. These packets may be dropped if the SCP is overloaded. Some buffer quotas, such as those for ICC packets, can be sized so that packets are never dropped. Since software is not involved in the decision to preserve important packets or to drop excessive loads, high throughput is maintained during periods of overload. In practice, when the network topology is stable, the SCP is not overloaded and packets passing through the SCP for bridging are not dropped, even on networks with thousands of stations. This feature is most important during power-up or topology-change transients, to ensure the network progresses to the stable state.

If the SCP simply processed packets in FIFO order, reception of each packet type would be ensured, but timely processing of important packets might not. Therefore, the first step in any packet processing is to enqueue the packet for later processing. (Packets may be fully processed and the buffers reclaimed if the amount of work to do is no greater than the enqueue/dequeue overhead.) Since the operating system scheduler services each queue in turn, splitting into multiple queues allows the important packets to bypass the less important packets.

Multiple queues are also used on the output port of the SCP. These software output queues are serviced to produce a hardware output queue that is long enough to amortize device driver entry overheads, yet short enough to bound the service time for the last packet inserted. Bounding the hardware queue service time ensures that the important 802.1d control packets convey timely information for the distributed spanning tree algorithms. These considerations yield the queuing diagram shown in Figure 3.

At time t1, packets arriving in nonempty quotas are transferred by direct memory access (DMA) into

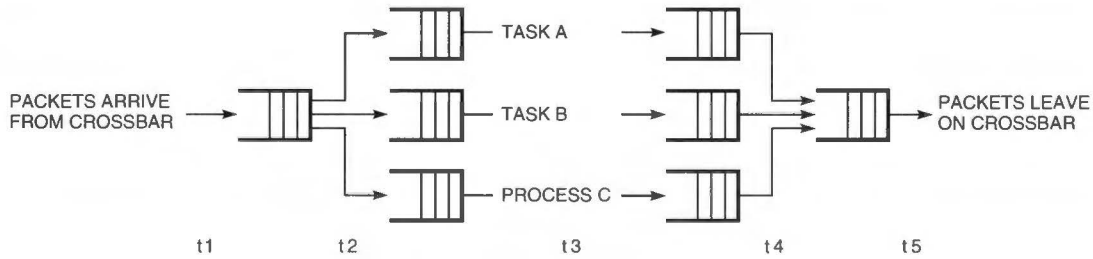


Figure 3 Packet Queuing on the SCP

dynamic RAM. They enter the hardware-received packet queue. At time t2, software processes the received packet queue, limiting the per-packet processing to simple actions like the enqueuing of the packet to a task or process. At time t3, the packet contents are examined and the proper protocol actions executed. This may involve the forwarding of the arriving packet or the generation of new packets. At time t4, packets are moved from the software output queues to the short hardware output queue. At time t5, the packet is transferred by DMA into the crossbar.

Limiting Malicious Influences Using packet types and buffer quotas, the SCP can distinguish important traffic, like bridge control messages, when it is subjected to an overload of bridge control, unknown destination addresses, and multicast messages. Such simple distinctions would not, however, prevent a malicious station from consuming all the buffers for multicast packets and allowing starvation of multicast-based protocols. Some of these protocols, like the IP address resolution protocol (ARP), become important when they are not allowed to function.[6]

To address this problem, the SCP also uses the incoming port to classify packets. A malicious station can wreak havoc on its own LAN whether or not the GIGAswitch system is present. By classifying packets by incoming port, we guarantee some buffers for each of the other interfaces and thus ensure communication among them. The malicious station is reduced to increasing the load of nuisance background traffic. Region t4 of Figure 3 contains the layer of flooding output queues that sort flooded packets by source port. When forwarding is done by the SCP bridge code, packets from well-behaved networks can bypass those from poorly behaved networks.

Fragmentation of resources introduced by the fine-grained packet classification could lead to small buffer quotas and unnecessary packet loss. To compensate for these possibilities, we provided shared resource pools of buffers and high-throughput, low-latency packet forwarding in the SCP.

Guaranteeing Forward Progress If an interrupt-driven activity is offered unlimited load and is allowed to attempt to process the unlimited load, a "livelock" condition, where only that activity executes, can result. Limiting the rate of interrupts allows the operating system scheduler access to the CPU, so that all parts of the system can make forward progress in a timely manner.

On the SCP, limiting the interrupt rate is accomplished in two ways. One is to mask the propagation of an interrupt by combining it with a software-specified pulse. After an interrupt is serviced, it is inhibited for the specified time by triggering the start of the pulse. At the cost of hardware complexity, software is given a method for quick, single-instruction, fine-grained rate limiting. Another method, suitable for less frequently executed code paths like error handling, is to use software timers and interrupt mask registers to limit the frequency of an interrupt. Limiting the interrupt rate also has the beneficial effect of amortizing interrupt overhead across the events aggregated behind each interrupt.

Noninterrupt software inhibits interrupts as part of critical section processing. If software inhibits interrupts for too long, interrupt service code cannot make forward progress. By convention, interrupts are inhibited for a limited time.

Interrupt servicing can be divided into two types. In the first type, a fixed sequence of actions is taken, and limiting the interrupt rate is sufficient to limit interrupt execution time. Most error processing falls into this category. In the second type, for all practical purposes, an unbounded response is required. For example, if packets arrive faster than driver software can process them, then interrupt execution time can easily become unacceptable. Therefore, we need a mechanism to bound the service time. In the packet I/O interrupt example, the device driver polls the microsecond clock to measure service time and thereby terminate device driver processing when a bound is reached. If service is prematurely terminated, then the hardware continues to post the interrupt, and service is renewed when the rate-limiting mechanism allows the next service period to begin.

Interrupt rate limiting can lead to lower system throughput if the CPU is sometimes idle. This can be avoided by augmenting interrupt processing with polled processing when idle cycles remain after all activities have had some minimal fair share of the CPU.

Reliability and Availability

Network downtime due to switch failures, repairs, or upgrades of the GIGAswitch system is low. Components in the GIGAswitch system that could be single points of failure are simple and thus more

reliable. Complex functions (logic and firmware) are placed on modules that were made redundant. A second SCP, for example, takes control if the first SCP in the GIGAswitch system fails. If a LAN is connected to ports on two different FDDI line cards, the 802.1d spanning tree places one of the ports in backup state; failure of the operational port causes the backup port to come on-line.

The GIGAswitch system allows one to "hot-swap" (i.e., insert or remove without turning off power) the line cards, SCP, power supply front-ends, and fans. The GIGAswitch system may be powered from station batteries, as is telephone equipment, to remove dependency on the AC mains.

Module Details

In the next section, we describe the functions of the clock card, the switch control processor, and the FDDI line card.

Clock Card

The clock card generates the system clocks for the modules and contains a number of centralized system functions. These functions were placed on the clock card, rather than the backplane, to ensure that the backplane, which is difficult to replace, is passive and thus more reliable. These functions include storing the set of 48-bit IEEE 802 addresses used by the switch and arbitration of the backplane bus that is used for connection setup.

Management parameters are placed in a stable store in flash electrically erasable programmable read-only memory (EEPROM) on the clock card, rather than on SCP modules. This simplifies the task of coordinating updates to the management parameters among the SCP modules. The SCP module controlling the box is selected by the clock card, rather than by a distributed (and more complex) election algorithm run by the SCPs.

The clock card maintains and distributes the system-wide time, communicated to the SCP and line cards through shared-memory mailboxes on their GPI chips. Module insertion and removal are discovered by the clock card, which polls the slots in the backplane over the module identification bus interconnecting the slots. The clock card controls whether power is applied to or removed from a given slot, usually under command of the SCP. The clock card also provides a location for the out-of-band management RS-232 port, although the out-of-band management code executes on the SCP.

Placing this set of functionality on the clock card does not dramatically increase complexity of that module or reduce its reliability. It does, however, significantly reduce the complexity and increase the reliability of the system as a whole.

Switch Control Processor

The SCP module contains a MIPS R3000A CPU with 64-kilobyte (kB) instruction and data caches, write buffer, 16-megabyte (MB) DRAM, 2-MB flash memory, and crossbar access control hardware. Figure 4 shows a diagram of the SCP. The large DRAM provides buffering for packets forwarded by the SCP and contains the switch address databases. The XAC provides robust operation in the face of overload and an efficient packet flooding mechanism for

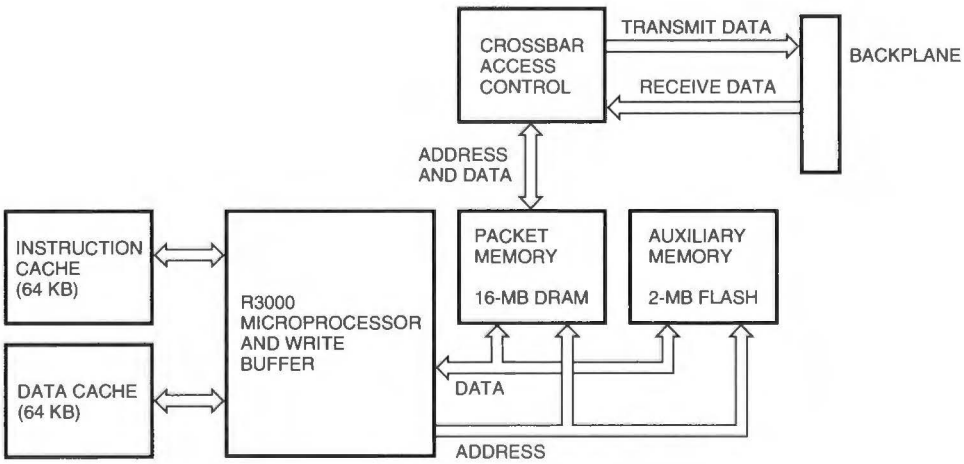


Figure 4 Switch Control Processor

highly populated GIGAswitch systems. The XAC is implemented using two field-programmable gate arrays with auxiliary RAM and support logic. A major impetus for choosing the MIPS processor was software development and simulation tools available at the time.

We input address traces from simulations of the SCP software to a cache simulation that also understood the access cost for the various parts of the memory hierarchy beyond the cache. Using the execution time predicted by the cache simulation, we were able to evaluate design trade-offs. For compiler-generated code, loads and stores accounted for about half the executed instructions; therefore cache size and write buffer depth are important. For example, some integrated versions of the MIPS R3000 processor would not perform well in our application. Simulation also revealed that avoiding stale data in the cache by triggering cache refills accounted for more than one-third of the data cache misses. Consequently, an efficient mechanism to update the cache memory is important. It is not important, however, that all DMA input activity updates the cache. A bridge can forward a packet by looking at only small portions of it in low-level network headers.

Cache simulation results were also used to optimize software components. Layers of software were removed from the typical packet-processing code paths, and some commonly performed operations were recoded in assembly language, which yielded tighter code and fewer memory references.

The conventional MIPS method for forcing the cache to be updated from memory incurs three steps of overhead.[7] One step is linear in the amount of data to be accessed, and the other two are inefficient for small amounts of data. During the linear time step, the information to be invalidated is specified using a memory operation per tag while the cache is isolated from memory. This overhead is avoided on the SCP by tag-bit manipulations that cause read memory operations to update the cache from DRAM. No additional instructions are required to access up-to-date information, and the method is optimal for any amount of data.

FDDI Line Card

The FGL contains one FDDI port subsystem per port (two for FGL-2 and four for FGL-4) and a processor subsystem. The FDDI port systems, shown in Figure 5, are completely independent. The process for sending a packet from one FDDI LAN to another is the same, whether or not the two FDDI ports are on the same FGL module or not. The processor subsystem consists of a Motorola 68302 microprocessor with 1 MB of DRAM, 512 kB of read-only memory (ROM), and 256 kB of flash memory. The processor subsystem is used for initial configuration and setup, diagnostics, error logging, and firmware functions. The FGL reused much firmware from

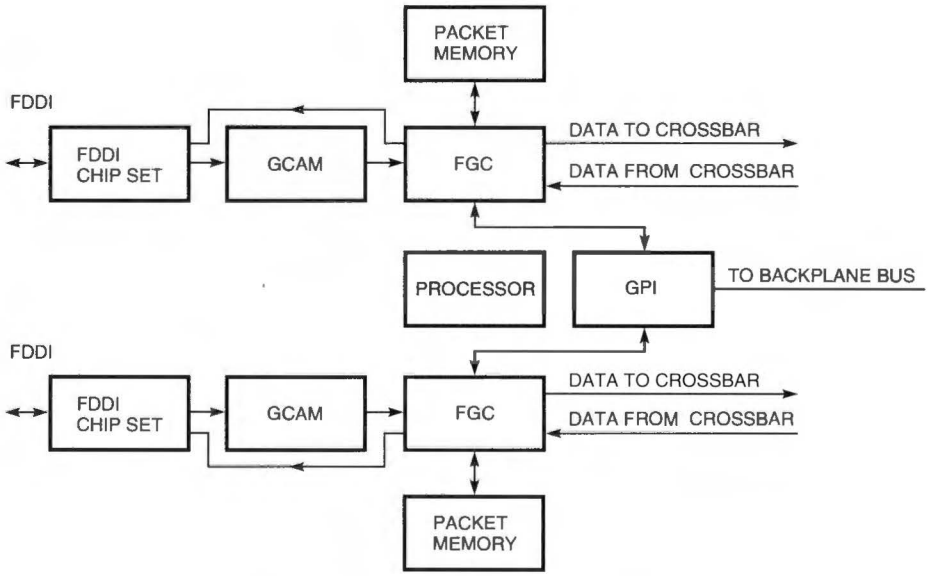


Figure 5 FDDI Line Card for Two Ports

other network products, including the DECNIS 600 FDDI line card firmware; station management functions are provided by the Common Node Software.[8]

The FGL uses daughter cards, called ModPMDs, to implement a variety of physical media attachments, including single- and multiple-mode fiber usable for distances of up to 2 and 20 kilometers, respectively, and unshielded, twisted-pair (UTP) copper wire, usable up to 100 meters. Both the FGL-2 and the FGL-4 can support four ModPMDs. The FGL-2 can support one or two attachments per FDDI LAN and appear as a single attachment station (SAS), dual attachment station (DAS), or M-port, depending upon the number of ModPMD cards present and management settings. The FGL-4 supports only SAS configurations. The Digital VLSI FDDI chip set was used to implement the protocols for the FDDI physical layer and MAC layer. Each FDDI LAN can be either an ANSI standard 100 Mb/s ring or a full-duplex (200 Mb/s) point-to-point link, using Digital's full-duplex FDDI extensions.[9]

The heart of the FGL is the bridge forwarding component, which consists of two chips designed by Digital: the FDDI-to-GIGAswitch network controller (FGC) chip and the GIGAswitch content addressable memory (GCAM) chip. It also contains a set of medium-speed static RAM chips used for packet storage and lookup tables.

The GCAM provides a 256-entry associative memory for looking up various packet fields. It is used to match 1-byte FDDI packet control fields, 6-byte destination and source addresses, 1-byte destination service access point (DSAP) fields, and 5-byte subnetwork access protocol service access point (SNAP SAP) protocol identifiers. The GCAM chip has two data interfaces: a 16-bit interface used by the processor and an 8-bit, read-only interface that is used for on-the-fly matching of packet fields. The FGC can initiate a new lookup in GCAM every 80 nanoseconds.

The FGC chip is a large (approximately 250,000 transistors), 240-pin, 1.0-micrometer gate array that provides all the high-performance packet queuing and forwarding functions on an FGL. It also controls the packet flow to and from the crossbar and to and from the FDDI data link chip set. It queues inbound and outbound packets, splitting them by type into queues of a size determined by firmware at start-up time. The FGC looks up various FDDI packet fields (1) to determine whether or not to forward a packet and which port to use, (2) to determine whether the packet contains a new source address and to note when each source address was last heard, and (3) to map packet types to classes used for filtering. It also provides a number of packet and byte counters. The FGC chip is also used in the FDDI line card for the DECNIS multiprotocol router.

FDDI Bridge Implementation

In the next section, we describe the implementation of an FDDI bridge on the GIGAswitch system platform.

Packet Flow

Figure 6 shows the interconnection of modules in a GIGAswitch system. Only one port of each FDDI line card is shown; hardware is replicated for other ports. Note that the line cards and SCP each have dual-simplex 100 Mb/s connections to the crossbar; traffic can flow in both directions simultaneously.

The FGC hardware on the GIGAswitch system line card looks up the source address, destination addresses, and protocol type (which may include the frame control [FC], DSAP, SNAP SAP, etc., fields) of each packet received. The result of the lookup may cause the packet to be filtered or dropped. If the packet is to be forwarded, a small header is prepended and the packet is placed on a queue of packets destined for the crossbar. Buffers for bridge control traffic are allocated from a separate pool so that bridge control traffic cannot be starved by data traffic. Bridge control traffic is placed in a separate queue for expedient processing.

Most packets travel through the crossbar to another line card, which transmits the packet on the appropriate FDDI ring. If the output port is free, the GIGAswitch system will forward a packet as soon as enough of the packet has been received to make a forwarding decision. This technique, referred to as cut-through forwarding, significantly reduces the latency of the GIGAswitch system.

Some packets are destined for the SCP. These are network management packets, multicast (and broadcast) packets, and packets with unknown destination addresses. The SCP is responsible for coordinating the switch-wide resources necessary to forward multicast and unknown destination address packets.

Learning and Aging

A transparent bridge receives all packets on every LAN connected to it and notes the bridge port on which each source address was seen. In this way,

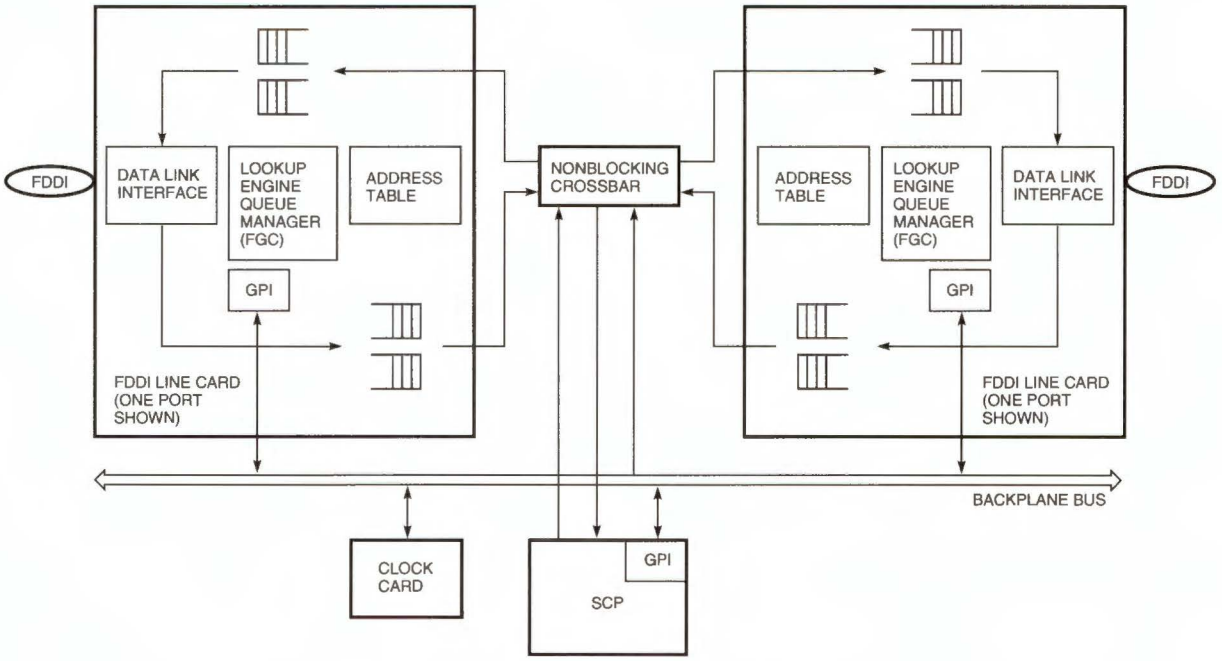


Figure 6 GIGAswitch System Modules

the bridge learns the 48-bit MAC addresses of nodes in a network. The SCP and line cards cooperate to build a master table of 48-bit address-to-port mappings on the SCP. They maintain a loose consistency between the master address table on the SCP and the per-port translation tables on the line cards.

When a line card receives a packet from a source address that is not in its translation table, it forwards a copy of the packet to the SCP. The SCP stores the mapping between the received port and the 48-bit address in the master address table and informs all the line cards of the new address. The SCP also polls the line cards at regular intervals to learn new addresses, since it is possible for the packet copy to be dropped on arrival at the SCP. The FGC hardware on the line card notes when an address has been seen by setting a source-seen bit in the addresses' translation table entry.

Since stations in an extended LAN may move, bridges remove addresses from their forwarding tables if the address has not been heard from for a management-specified time through a process called aging. In the GIGAswitch system, the line card connected to a LAN containing an address is responsible for aging the address. Firmware on FGL scans the translation table and time-stamps entries that have the source-seen bit set. A second firmware task scans the table, placing addresses that have not been seen for a specified time on a list that is retrieved at regular intervals by the SCP. Aged addresses are marked but not removed from the table unless it is full; this reduces the overhead if the address appears again.

The GIGAswitch system should respond quickly when an address moves from one port of the switch to another. Many addresses may move nearly simultaneously if the LAN topology changes. The fact that an address is now noticed on a new port can be used to optimize the address aging process. A firmware task on FGL scans the translation table for addresses that have been seen but are not owned by this port and places them on a list. The SCP then retrieves the list and quickly causes the address to be owned by the new port.

Multicast to Multiple Interfaces

The SCP, rather than the line cards, sends a packet out multiple ports. This simplifies the line cards and provides centralized information about packet flooding in order to avoid overloading remote lower-speed LANs with flooded packets. The SCP allows network management to specify rate limits for unknown destinations and for multicast destination traffic.

Flooding a packet requires replicating the packet and transmitting the replicas on a set of ports. During the design of the flooding mechanism, we needed to decide whether the replication would take place on the SCP, in hardware or software, or on the line cards. The design criteria included (1) the amount of crossbar bandwidth that is consumed by the flooding and (2) the effect of the flooding implementation on the forwarding performance of unicast packets.

The size of the switch and the filters that are specified also affected this decision. If the typical switch is fully populated with line cards and has no filters set, then one incoming multicast packet is flooded out a large number of ports. If a switch has few cards and filters are used to isolate the LANs, then an incoming multicast packet may not have to be sent out any port.

Since the crossbar design allows many outputs to be connected to a single input, a single copy of a packet sent into the crossbar can be received by multiple FDDI ports simultaneously. This improves the effective bandwidth of the SCP connection to the crossbar since fewer iterations of information are required. The queuing mechanisms used to send multicast connection information over the backplane allow each line card port using the multicast crossbar facility to receive a copy no more than one packet time after it is ready to do so.

We decided to implement flooding using special hardware on the SCP. The DMA transfers the packet once into a private memory, and all iterations proceed from that memory, thus removing contention at the SCP's DRAM. The multicast hardware repeatedly transmits the packet into the crossbar until the backplane queuing mechanisms signal that all relevant ports have received a copy.

Integration and Test

GIGAswitch system software development and test were performed in parallel with hardware development. Most of the SCP software was developed before reliable SCP hardware was available and before integration of the various modules could proceed in a GIGAswitch system. The simulation of hardware included the SCP's complete memory map, a simplified backplane, a simplified clock card, and FDDI line cards. At run time, the programmer could choose between line card models connected to real networks or real GIGAswitch system FGLs. Instruction and data address references were extracted from the instruction interpreter that understood the SCP memory map.

FGL testing initially proceeded in standalone mode without other GIGAswitch system modules. The FGL firmware provided a test mechanism that allowed ICC packets to be received and transmitted over a serial port on the module. This facility was used to test ICC processing by sending ICC packets from a host computer over a serial line. The next level of testing used pairs of FGLs in a GIGAswitch system backplane. To emulate the SCP hardware, one FGL ran debug code that sent ICC packets through the crossbar.

Initial testing of SCP/FGL interaction used the FGL's serial ICC interface to connect an FGL to SCP software emulated on workstations. This interface allowed SCP and FGL firmware to communicate with ICC packets and permitted testing of SCP and FGL interactions before the SCP hardware was ready.

Network Management

The GIGAswitch system is manageable via the SNMP protocol. SNMP uses get and get-next messages to examine and traverse the manageable objects in the GIGAswitch system. SNMP uses set messages to control the GIGAswitch system.

A single SNMP set message can manipulate multiple objects in the GIGAswitch system. The objects should change atomically as a group, with all of them modified or none of them modified from the viewpoint of the management station. If the management station indicates that the GIGAswitch system reported an error, there are no dangling side effects. This is most advantageous if the management station maps a single form or command line to a single SNMP set message.

The SCP software achieves atomicity by checking individual object values, cross-checking object values, modifying object values with logging, and recording commit/abort transactional boundaries in a phased process. As each object value is modified, the new value is logged. If all modifications succeed, a commit boundary is recorded and the SNMP reply is sent. If any modification fails, all preceding modifications for this set operation are rolled back, an abort boundary is recorded, and the SNMP reply is sent.

Measured Performance

Measuring the performance of a GIGAswitch system requires a significant amount of specialized

equipment. We made our performance measurements using proprietary FDDI testers constructed at Digital. Under the control of a workstation, the testers can send, receive, and compare packets at the full FDDI line rate. We used 21 testers in conjunction with commercial LAN analyzers to measure performance of the GIGAswitch system.

The forwarding rate was measured by injecting a stream of minimum-size packets from a single input port to a single output port. This test yields a forwarding rate of 227,000 packets per second. The forwarding rate in this test is limited because connection requests are serialized when the (single) output port is busy. The arbitration mechanism allows connections to ports that are not busy to be established in parallel with ongoing packet transmissions. Modifying the test so that one input port sends packets to three output ports increases the aggregate forwarding rate to 270,000 packets per second. We also measured the aggregate forwarding rate of a 22-port GIGAswitch system to be approximately 3.8 million minimum-sized FDDI packets per second. At very high packet rates, small differences in internal timing due to synchronization or traffic distribution can exaggerate differences in the forwarding rate. The difference between 270,000 packets per second and 227,000 packets per second is less than 9 byte times per packet at 100 Mb/s.

For the GIGAswitch system, the forwarding latency is measured from first bit in to first bit out of the box. Forwarding latency is often measured from last bit in to first bit out, but that method hides any delays associated with packet reception. The forwarding latency was measured by injecting a stream of small packets into the switch at a low rate, evenly spaced in time. The forwarding latency was determined to be approximately 14 microseconds, or approximately 175 byte times at 100 Mb/s. This measurement illustrates the result of applying distributed, dedicated hardware to the forwarding path. It includes two 48-bit addresses and a protocol type lookup on the incoming line card, the output filtering decision on the outbound line card, and the delays due to connection latency, data movement, and synchronization.

The GIGAswitch system filters minimum-sized packets from a multiple-access FDDI ring at the line rate, which is approximately 440,000 packets per second per port. Filtering is necessary to reject traffic that should not be forwarded through the switch.

By design, the GIGAswitch system flooding rate is lower than the rate for unicast traffic. We measured the flooding rate to be 2,700 packets per second. Note that a multicast rate of r implies transmission of $(s \times r)$ packets by the switch, where s is the number of ports currently on the bridge spanning tree.

Measurement of GIGAswitch system flooding rates on real networks within Digital and at field test sites indicates that the switch is not a bottleneck for multicast traffic.

Conclusions

The GIGAswitch system is a general-purpose, packet-switching platform that is data link independent. Many issues were considered and techniques were used in its design to achieve robustness and high performance. The performance of the switch is among the highest in the industry. A peak switching rate of 6.25 million variable-size packets per second includes support for large hunt groups with no loss in performance. Digital's first product to include a GIGAswitch system was a 22-port IEEE 802.1d FDDI bridge. Shipment to general customers started in June 1993. A four-port FDDI line card (FGL-4) is in development. A two-port GIGAswitch system line card (AGL-2) using ATM is planned for shipment in May 1994. This card uses modular daughter cards to provide interfaces to synchronous optical network (SONET) STS-3c, synchronous digital hierarchy (SDH) STM-1, DS-3, or E3 transmission media.

The GIGAswitch system can be extended to include other data links such as high-speed serial interface (HSSI), fast Ethernet, and Fibre Channel. High-performance routing is also possible.

The GIGAswitch system has been used successfully in a wide variety of application areas, including workstation farms, high-energy physics, and both LAN and metropolitan area network (MAN) backbones. One or more GIGAswitch systems can be used to construct a large-scale WAN backbone that is surrounded by routers to isolate individual LANs from the WAN backbone. A GIGAswitch system network can be configured to provide the bandwidth needed to support thousands of conventional LAN users as well as emerging applications such as large-scale videoconferencing, multimedia, and distributed computing.

Acknowledgment

Our thanks to the following engineers who contributed to the GIGAswitch system architecture and product: Vasmi Abidi, Dennis Ament, Desireé

Awiszio, Juan Becerra, Dave Benson, Chuck Benz, Denise Carr, Paul Clark, Jeff Cooper, Bill Cross, John Dowell, Bob Eichmann, John Forecast, Ernie Grella, Martin Griesmer, Tim Haynes, Chris Houghton, Steve Kasischke, Dong Kim, Jeanne Klauser, Tim Landreth, Rene Leblanc, Mark Levesque, Michael Lyons, Doug MacIntyre, Tim Martin, Paul Mathieu, Lynne McCormick, Joe Mellone, Mike Pettengill, Nigel Poole, Dennis Racca, Mike Segool, Mike Shen, Ray Shoop, Stu Soloway, Dick Somes, Blaine Stackhouse, Bob Thibault, George Varghese, Greg Waters, Carl Windnagle, and Mike Wolinski.

References

1. *Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges* (New York: Institute of Electrical and Electronics Engineers, Inc., IEEE Standard 802.1D-1990).

2. A. Pattavina, "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE Journal on Selected Areas in Communication,* vol. 6, no. 9 (December 1988): 1489–1499.

3. W. Gilbert, *Modern Algebra with Applications* (New York: John Wiley, 1976).

4. M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communication,* vol. COM-35, no. 12 (December 1987): 1347–1356.

5. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," *Internet Engineering Task Force,* RFC 1157 (August 1990).

6. D. Plummer, "An Ethernet Address Resolution Protocol," *Internet Engineering Task Force,* RFC 826 (November 1982).

7. G. Kane, *MIPS RISC Architecture* (Englewood Cliffs, NJ: Prentice-Hall, 1989).

8. P. Ciarafella, D. Benson, and D. Sawyer, "An Overview of the Common Node Software," *Digital Technical Journal,* vol. 3, no. 2 (Spring 1991): 42–52.

9. H. Yang, B. Spinney, and S. Towning, "FDDI Data Link Development," *Digital Technical Journal,* vol. 3, no. 2 (Spring 1991): 31–41.

Lucien A. Dimino
Rabah Mediouni
T. K. Rengarajan
Michael S. Rubino
Peter M. Spiro

Performance of DEC Rdb Version 6.0 on AXP Systems

The Alpha AXP family of processors provided a dramatic increase in CPU speed. Even with slower processors, many database applications were dominated by relatively slow I/O rates. To maintain a balanced system, database software must incorporate techniques that specifically address the disparity between CPU speed and I/O performance. The DEC Rdb version 6.0 database management system contains shorter code paths, fewer I/O operations, and reduced stall times. These enhancements minimize the effect of the I/O bottleneck and allow the AXP processor to run at its intended higher speeds. Empirical performance results show a marked improvement in I/O rates.

The DEC Rdb for OpenVMS AXP product (hereafter in this paper designated as DEC Rdb) is Digital's flagship database management system.[1] The DEC Rdb relational database software competes effectively in multiple data processing domains such as stock exchanges, image-processing applications, telemedicine, and large databases used for decision support or scientific applications. Virtually all these application frameworks are demanding increased processing power and increased I/O capabilities.

The Alpha AXP processor family represents a quantum jump in the processing power of CPUs. It is designed to scale up to 1,000 times the current processing power in a decade.[2] On the other hand, disk I/O latency is improving at a much slower rate than CPU power. As a result, especially on an AXP platform, the total time to execute a query is dominated by the time to perform the disk I/O operations. This disparity between processor speed and I/O latency is commonly called the I/O bottleneck or the I/O gap.[3]

In this paper, we describe our efforts to improve the performance of DEC Rdb on AXP systems. First we explain general porting steps that ensure a foundation of good performance on Alpha AXP systems. Then we describe our efforts to reduce the I/O bottleneck. We present the performance enhancements to various components of earlier versions of DEC Rdb and compare the enhancements and new features of version 6.0. Finally, we discuss the TPC-A transaction processing benchmark and present empirical results that quantify the benefits of the optimizations.

Performance Gains During the Port to OpenVMS

In this section, we recount some of the initial, general performance modifications to the DEC Rdb software in the port from the VAX VMS system to the OpenVMS AXP system. We briefly discuss data alignment and reduction of the software subroutines or PALcode calls.

The DEC Rdb engineering developers saw the opportunity for increased performance through careful alignment and sizing of data structures. We needed to develop a solution that would exploit the performance gain of data alignment, yet still maintain an easy migration path for our large base of customers on VAX systems.

For the first release of DEC Rdb, all in-memory data structures were naturally aligned. In addition, many in-memory byte and word fields in these data structures were expanded to 32 bits of data (longwords). Once the in-memory data structures were aligned, we turned our attention to the on-disk data structures. The database root file, which is also frequently accessed, was completely aligned. New databases can be created with these aligned data structures, and existing databases can be aligned during a database convert operation on both VAX and AXP systems. This operation takes only seconds.

We did not align the data on the database pages in the storage area. Database pages contain the actual user data records. By leaving this data unaligned, we did not force a database unload/reload requirement on our customers. This factor and our support

of clusters for both the VAX and the Alpha AXP architectures simplify migration from the VAX system to the AXP system.

After we completed the port of DEC Rdb, we ran performance benchmarks to determine which areas of the system could be enhanced. By using an internal tool called IPROBE, we learned that we could improve the performance of DEC Rdb by rewriting or eliminating code paths in PALcode subroutines. From January 1993 through July 1993, we reduced PALcode cycles from 143k to 62.6k per TPC-A transaction. Details of earlier performance modifications have been discussed in this *Journal*.[4]

Performance Enhancements

In this section, we describe how we improved the performance of DEC Rdb by addressing I/O bottleneck and problems in the code path.

The general strategy to combat the I/O gap in DEC Rdb was twofold. The first step was to minimize I/O operations. We used the "global buffers" of DEC Rdb to avoid read I/O requests. We took advantage of the large physical memory available in modern computers to cache interesting parts of the database in memory and hence reduce the disk read I/O requests. The 64-bit Alpha AXP architecture allows computers to easily use more than 4 gigabytes (GB) of directly addressable memory. Write I/O requests are avoided by using the fast commit feature of DEC Rdb. The minimization of I/O activity is detailed in a technical report by Lomet et al.[5]

The second step was to reduce the stall time for the I/O operations that must be done. A reduced stall time allows DEC Rdb software to continue processing the queries even when disk I/O operations are in progress on its behalf. Two features of DEC Rdb version 6.0, the asynchronous prefetch and the asynchronous batch writes, reduce the stall time for the read and write I/O requests, respectively.

Although this strategy handles the I/O gap, the write to the after image journal (AIJ) becomes a limiting factor in high-performance transaction processing (TP) systems. The stall time for the AIJ write is reduced through the AIJ log server and AIJ cache on electronic disk features of DEC Rdb version 6.0.

In the following sections, we discuss these features in detail.

Write I/O Requests

To read a new set of pages from the database on disk, the DEC Rdb software selects a buffer in the buffer pool to be replaced. We refer to this as the victim buffer.

Prior to DEC Rdb version 6.0, writes of updated database pages to disk happened in synchronous batches. If the victim buffer is marked, a synchronous batch write is launched. In addition to the victim buffer, a number of least recently used (LRU) marked buffers are collected. The number of buffers in the batch write is specified as the BATCH_MAX parameter by users.

The list is then sorted by page numbers in order to perform global disk head optimization. After this, asynchronous disk write I/O requests are issued for all the marked buffers. In these earlier versions, the DEC Rdb product then waits until all the write I/O requests are completed. Although the individual writes are issued asynchronously and may complete in parallel, DEC Rdb waits synchronously for the entire batch write to complete. No query processing happens during the batch write. Figure 1 shows alternating synchronous batch writes and other work. Writes to the same disk are executed one after another, and writes to different disks are executed in parallel.

The synchronous batch leverages two important optimizations: (1) parallelism between various disks and (2) disk head optimizations implemented at various levels of the storage hierarchy. The synchronous batch write feature reduced the average stall time per disk write I/O. The extent of reduction depends upon the degree to which the above two optimizations happen, which in turn depends upon the physical database design and application query behavior. In the TPC-A benchmark, the synchronous batch writes reduced the average stall time for the account write by 50 percent compared to synchronous individual writes.

To further reduce stall time, we implemented asynchronous batch writes (ABWs) in DEC Rdb version 6.0. With ABW, DEC Rdb now maintains the last few buffers unmarked. The size of this clean region of the buffer pool is specified by the user. As new

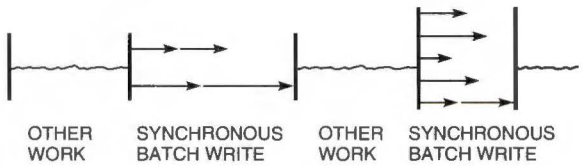


Figure 1 Alternating Synchronous Batch Writes and Other Work

pages are read from the database, database buffers migrate toward the end of the LRU chain of buffers. If a marked buffer were found in the clean region, an ABW would be invoked.

Figure 2 shows asynchronous batch writes invoked periodically, while other work continues. Processing does not explicitly wait for any of the disk write I/O requests to complete. (There may be implicit waits due to disk queuing effects.) Instead, processing continues. If a buffer with a pending write is chosen as the victim or if one of the buffers with pending writes is required for further processing, DEC Rdb then waits for completion of the pending writes. For applications with good temporal locality, it is likely that the buffers with pending writes will not be required for further processing.

Figure 2 also shows a rare instance in which other processing stops, waiting for one of the asynchronous write I/O requests to complete. Again, other processing includes stalls for disk read I/O requests. It is also possible to start a new ABW when the previous one has not yet completed.

Read I/O Requests

Requests for database pages are often satisfied by a large global buffer pool. This is true when the whole database fits in a large memory, common on Alpha AXP systems. Under certain circumstances, however, the buffer pool is not large enough to satisfy all requests. Moreover, seldom-used data may be replaced by more frequently used data in the buffer pool.

The essential strategy is to submit asynchronous disk read I/O requests well before the data is really needed during query processing. If the asynchronous prefetch (APF) request is made far enough in advance of the actual request, the process will not need to stall for the I/O. Critical to any prefetch strategy is to reliably determine the desirable database pages for the immediate future.

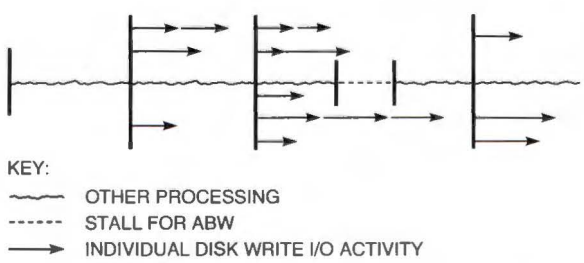


Figure 2 Simultaneous Asynchronous Batch Writes and Other Work

Fortunately, applications request data in sets of rows. Therefore, based on user requests and the query optimizer decisions, the database access patterns are known at the time query execution starts. This allows the mechanism to prefetch the data from the database into the buffer pool.

In DEC Rdb version 6.0, we implemented the mechanism to prefetch data from the database based on requests from higher layers of DEC Rdb. This is performed in an integrated manner in the buffer pool with the usual page locking protection in a cluster. We also implemented the policy to use asynchronous prefetch in the case of sequential scans.

Sequential scans are quite common in databases for batch applications producing large reports. They are also chosen by the optimizer when a large number of records are selected from a table for a query for processing.

The user can specify the parameter APF_DEPTH that controls the number of buffers to be used for prefetching database pages, hence the lead of prefetch ahead of the real fetches. With a sufficient level of prefetch, it is quite possible to exploit the parallelism in the disk subsystem as well as achieve close to the spiral transfer rate of disks, as we describe in the following test.

We placed one table in a mixed format area on an RA73 disk. The database server process used 400 buffers of 6-kilobyte (kB) size, and the APF_DEPTH parameter was set to 50 buffers. With these settings, the sequential scan of a 1-GB area took 593 seconds. This is equivalent to a transfer rate of 1.69 megabytes per second (MB/s), compared to the rated spiral transfer rate of 1.8 MB/s for the disk.

We performed another test to determine the improvement with the APF feature in DEC Rdb version 6.0. Again, we built a database with one table in a mixed format area, but this time on a stripe set of two RA92 disks. The database server process used 400 buffers of 3-kB size. The elapsed time to scan a 1-GB area in version 5.1 was 6,512 seconds, and the transfer rate was 0.15 MB/s. In version 6.0 the elapsed time was 569 seconds, and the transfer rate was 1.76 MB/s. An APF_DEPTH of 100 buffers was used for version 6.0. We find that APF has made sequential scan 10 times faster in DEC Rdb version 6.0. Note that this performance improvement can be made much better by using more disks in the stripe set and by using more powerful processors.

Commit Time Processing

Although the disk I/O stalls are significantly reduced by the APF and ABW in DEC Rdb version 6.0, the commit time processing remains a significant wait.

Prior to version 6.0, DEC Rdb software used a cooperative flushing protocol. Each database server produced the AIJ log records and then used the lock manager to determine the group committer. The group committer then formatted the AIJ log records for all the servers and flushed the data to the AIJ file in one disk write I/O. Each server thus competed for the AIJ lock in order to become the group committer. Each server also had to acquire the AIJ lock even to determine if it had committed. Figure 3 shows this algorithm. DEC Rdb software also supports timer-based tuning methods to increase the group size for group commit.[6]

AIJ Log Server With DEC Rdb version 6.0, a dedicated process called the AIJ log server (ALS) runs on every node of the cluster to perform the task of processing group writes to the AIJ file for all servers. The new algorithm is in two parts, one for the database servers and another for the ALS. Figure 4 shows these algorithms.

At commit time, database servers generate AIJ log records, store them in shared memory, and go to "sleep." They are "woken up" by the ALS when their commit records are successfully written to the AIJ file. The ALS gathers the AIJ log records of all users, formats them, writes them to the AIJ file, and then wakes up servers waiting for commit. Thus, the ALS performs AIJ writes in a continuous loop.

The ALS allows DEC Rdb version 6.0 to scale up to thousands of transactions per second with one magnetic disk using the TPC-A benchmark.

With the ALS, the average stall time for a server to commit is 1.5 times more than the time taken to perform one log write I/O. This stall is of the order of 17 milliseconds with 5,400-rpm disks. Note that this stall time is a function of disk performance only and is independent of the workload. In a high-throughput TP environment, where transaction times are very short, this stall at commit time is still a significant wait. Considering the speed of Alpha AXP processors, the commit stall is many times more than the processing time for servers.

AIJ Cache on Electronic Disk The AIJ cache on electronic disk (ACE) is a feature of the ALS that also helps high-throughput TP systems. ACE utilizes a small amount (less than 1 MB) of very low latency, solid-state disk to reduce the commit stall time. Figure 5 shows the new ACE algorithm with ALS. The ACE file is on a solid-state disk shared by all nodes of a cluster. The file is partitioned for use by various ALS servers in the cluster.

```
Put AIJ data in shared memory

Get AIJ lock

if      our data IS flushed
then    begin

        Release AIJ lock

        return
        end

! We are the group committer

Format AIJ data in shared memory

Reserve space in AIJ file

Release AIJ lock

Write AIJ data to AIJ file

Indicate AIJ data is flushed for group members
```

Figure 3 Cooperative Flushing Algorithm

```
Put AIJ data in shared memory
Until   committed
        sleep       ! Woken up after commit by ALS
```

(a) Database Server Algorithm

```
Get AIJ lock
loop    begin
        Format AIJ data in shared memory
        Write AIJ data to AIJ file
        Indicate AIJ data is flushed for group members
        Wake up all committed processes
        end
Release AIJ lock
```

(b) ALS Algorithm

Figure 4 AIJ Log Server Process

```
Get AIJ lock
loop    begin
        Format AIJ data in shared memory
        Save AIJ block # and length of I/O
            in ACE header
        Write AIJ data and ACE header to ACE file
            ! 1ms
        Wake up all committed processes
        Write AIJ data to AIJ file    ! 11ms
        Set flags to indicate AIJ data is flushed
            for group members
        end
Release AIJ lock
```

Figure 5 ACE Algorithm

Basically, the data is first written to the ACE file, and the servers are allowed to proceed to the next transaction. The data is then flushed to the AIJ file in a second I/O. The write to the ACE disk includes the virtual block number of the AIJ file where the log data is supposed to be written. The ACE disk serves as a write-ahead log for the AIJ write. By doubling the number of disk writes per group, we reduced the response time for the servers to 0.5 times the stall for AIJ write and 1.5 times the stall for ACE write. This reduction in the stall time conversely increases the CPU utilization of servers, thereby reducing the number of servers required to saturate an AXP CPU.

Backup and Restore Operations

Simply stated, the way we traditionally perform backup does not scale well with system capacity, rarely uses its resources effectively, and is disruptive to other database activity. We designed the backup and restore operations to resolve these issues. Before we present that discussion, we first examine the problems with the traditional database backup operation.

Traditional Backup Process Using OpenVMS BACKUP as an example of the traditional backup process, we find that a single backup process performance does not scale with CPU performance, nor with system aggregate throughput. Instead, it is limited by device throughput. The only way to increase the performance is to perform multiple backups concurrently. However, concurrent backups executing on the same CPU interfere with one another to some extent. Each additional backup process provides less than 90 percent of the performance of the previous one. Five OpenVMS BACKUP operations executing on one CPU provide no greater performance than four operations, each executing on its own CPU. Consequently, five tape drives may provide only four times the performance of a single drive.

The OpenVMS BACKUP operation is limited by the lesser of the disk throughput and the tape throughput. Read performance for an RA73 disk may be as high as 1.8 MB/s, but OpenVMS BACKUP more typically achieves between 0.8 and 1.0 MB/s. Performance for a TA92 tape is 2.3 MB/s. Five tape drives, with an aggregate throughput of 11.5 MB/s, and 25 disks, with an aggregate throughput of 45.0 MB/s, can only be backed up at a rate of 4.0 MB/s (14.4 GB per hour).

Increasing the CPU capacity and aggregate throughput improves the backup performance but not proportionally. Low device utilization and nonlinear scaling mean that as the system capacity and database size increase, the cost in system time and hardware for a given level of backup performance becomes increasingly burdensome.

The traditional backup process, such as provided by OpenVMS BACKUP, is not coordinated with database activity. The database activity must be prohibited during the backup. If it is not, the restore operation will produce an inconsistent view of the data. In the latter case, database activity journals are required to return the database to consistency. Application of these journals significantly reduces the performance of the restore process.

To maximize the performance of the traditional backup process, the backup must consume a large portion of the entire throughput of the disks being backed up. As a result, database activity and backup activity severely impede each other when they compete for these disks.

RMU BACKUP Operation The RMU BACKUP operation resolves the problems with the traditional backup operation. RMU BACKUP is coordinated with database activity. It produces a consistent image of the database at a point in time without restricting database activity or requiring application of journals after the restore operation.

The RMU BACKUP operation is a multithreaded process; therefore, it backs up multiple disks to multiple tapes and eliminates the limiting factor associated with throughput of a single disk or a single tape drive. The aggregate of disk and tape throughputs determines the performance of RMU. Because the aggregate disk throughput is usually significantly higher than the aggregate tape throughput, all the disks have spare throughput at all times during RMU BACKUP. Consequently, the RMU BACKUP process and database activity interfere to a much lesser extent than is the case with traditional backup. Its multithreaded design also scales linearly with CPU capacity, aggregate disk throughput, and aggregate tape throughput.

The first steps of an RMU BACKUP are to evaluate the physical mapping of the database to disk devices and to determine the system I/O configuration. RMU then devises a plan to execute the backup. The goals of this plan are to divide the data among the tape drives equally, to prohibit interference between devices sharing a common I/O path, and to minimize disk head movement. The generated plan is a compromise because complete and accurate configuration data is difficult to collect

and costly to assimilate. To implement the plan, RMU creates a network of interacting threads. Each thread operates asynchronously, performing asynchronous multibuffered I/O to its controlled device. Interthread communication occurs through buffer exchange and shared memory structures.

RMU uses the database page checksum to provide end-to-end error detection between the database updater and the backup operation. It uses a backup file block cyclic redundancy check (CRC) to provide end-to-end error detection between the backup and the restore operations. In addition, RMU uses XOR recovery blocks to provide single error correction on restore. As a consequence, the data being backed up must be processed four times, which has a major effect on CPU usage. The first time is to evaluate the database page checksum. The second time is to copy the data and to exclude unused storage and redundant structures. Here we are willing to expend extra CPU cycles to reduce the I/O load. The third time is to generate an error detection code (CRC), and the fourth time is to generate an XOR error recovery block.

The RMU BACKUP operation improves performance in other ways. It does not back up redundant database structures, nor does it back up allocated storage that does not contain accessible data. As a result, the size of the backup file is significantly reduced, and relative backup performance is improved. The incremental backup feature selectively backs up only those database pages that have been modified. This provides an additional, significant reduction in backup file size relative to a file-oriented backup. These reductions in backup file size further improve RMU BACKUP performance relative to traditional backup.

The performance of the RMU RESTORE operation mirrors that of the RMU BACKUP operation. In spite of this, a natural asymmetry between the operations allows RMU BACKUP to outperform RMU RESTORE by 20 percent to 25 percent. There are several reasons for the asymmetry: the cost of allocating files, the asymmetry of read versus write disk performance, the asymmetric response of the I/O subsystem to read versus write I/O operations under heavy load, and the need to re-create or initialize redundant data that was not backed up by RMU BACKUP.

The RMU RESTORE operation can restore the entire database, selected database files, and even selected pages within the files. Restoring files or pages requires exclusive access to only the objects being restored. This is the only restriction on database activity during the restore operation.

We performed a test to illustrate the effectiveness and scalability of RMU BACKUP with database size and system capacity. Table 1 gives the results. This test also demonstrates the high level of backup performance that can be provided on AXP systems without the use of exotic or expensive technology. Although the results are limited by the aggregate tape I/O performance, the CPU does not appear to have sufficient excess capacity to justify a test with a sixth tape drive.

Unfortunately, comparable numbers are not available for other database products on comparable platforms. Nevertheless, when any database system relies on the operating system's backup engine, the expected performance limits are the same as the scenario presented.

Sorting Mechanism

A sorting mechanism is a key component of a database system. For example, the relational operations

Table 1 Backup Performance on AXP Systems

Configuration	Type	Amount
System	DEC 7000 Model 610 (182 MHz)	1
Disks	RA73	23
	RZ23	2
Controllers	HSC95	6
	CIXCD	6
	KMC44	5
Tape drive	TA92	5
Operating system	OpenVMS V1.5	
Database management software	DEC Rdb V5.1	
Database size of 48.8 GB (71% relational data or 34.7 GB)		
Performance		
Backup time	1:11:29	
Backup rate	41.0 GB/hour	
Sustained disk data rate of 455 kB/s per disk (25% utilization)		
Sustained tape data rate of 2.3 MB/s per tape (100% utilization)		
Restore time	1:32:49	
Restore rate	31.6 GB/hour	

of join, select, and project, which are executed to satisfy user queries, often sort records to enable more efficient algorithms. Furthermore, selected records must often be presented to the user in sorted order. Quite literally, a faster sorting mechanism directly translates to faster executing queries. Hence during the port to the Alpha AXP platform, we also focused on ensuring that the DEC Rdb sorting mechanism worked well with the Alpha AXP architecture. In the case of the sort mechanism, we reduced I/O stalls and optimized for processor cache hits rather than access main memory.

Prior to the port to Alpha AXP, DEC Rdb utilized only the replacement-selection sort algorithm. For AXP systems, we have modified the sort mechanism so that it can also utilize the "quicksort" algorithm under the right conditions.[7] In fact, we developed a modified quicksort that sorts (key prefix, pointer) pairs. Our patented modification allows a better address locality that exploits processor caching.[8] This is especially important on the Alpha AXP platform since the penalty for addressing memory is significant due to the relatively fast processor speed. The quicksort algorithm also performs better than the replacement-selection algorithm in a memory-rich environment, which is the trend for Alpha AXP systems.

To summarize, if the required sort fits in main memory, DEC Rdb version 6.0 utilizes the optimized quicksort algorithm; if the sort requires temporary results on disk, DEC Rdb uses the traditional replacement-selection sort algorithm. When writing temporary results to disk, both sort algorithms also utilize asynchronous I/O mechanisms to reduce I/O stalls.

The DEC Rdb version 6.0 implementation of the quicksort algorithm is based on the AlphaSort algorithm developed at Digital's San Francisco Systems Center. The AlphaSort algorithm achieved the world-record sort of 7 seconds on an industry-standard sort benchmark. This result is more than 3 times faster than the previous sort record of 26 seconds on a CRAY Y-MP system.[8]

Multi-statement Procedures

Prior to the implementation of multi-statement procedures in DEC Rdb, individual SQL statements were serially submitted to the database engine for execution. This method of execution incurs excessive code path because individual SQL statements must traverse different layers before they reach the database engine.

When clients and servers communicate over a network, each SQL statement incurs the additional overhead of two network I/O operations. The result is a long transaction code path as well as delays due to excessive network traffic. Figure 6 shows the execution path that individual SQL statements traverse in both the local and the remote cases.

Network overhead is a significant problem for client-server applications. Without the services of the VAX ACMS TP monitor, 14 network I/O operations are required to complete a single TPC-A transaction. Table 2 lists the TPC-A pseudocode needed to complete one transaction.

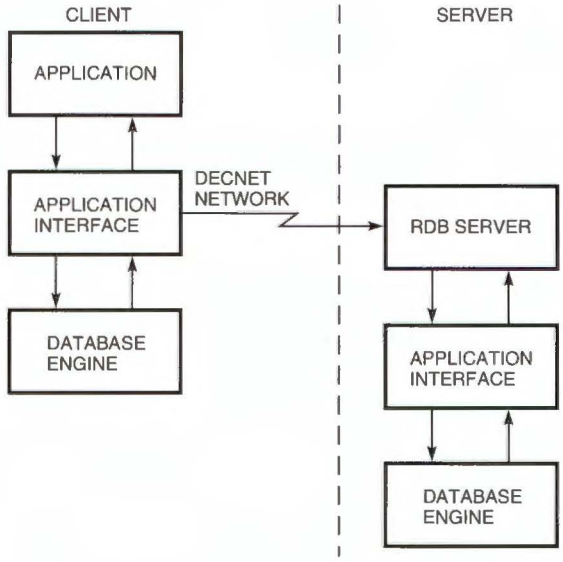


Figure 6 SQL Statement Execution Flow

Table 2 Individual SQL Statements for Single TPC-A Transaction

TPC-A Pseudocode	Network I/O Operations
`Start an update transaction`	2
`Update a row in BRANCH table`	2
`Update a row in TELLER table`	2
`Update a row in ACCOUNT table`	2
`Select Account_balance from ACCOUNT and display it on the terminal`	2
`Insert a row in the HISTORY table`	2
`Commit transaction`	2
Total network I/O operations per TPC-A transaction	14

Prior to DEC Rdb version 6.0, client applications accessing remote servers relied on the services of the VAX ACMS TP monitor to reduce the network overhead. With ACMS present on both the client and the server, a message carrying a transaction request is transferred to the server in one network I/O. As shown in Figure 7, an ACMS server is then selected to execute the request, and a message is returned to the client upon transaction completion. This reduces the number of network I/O operations to two per transaction. In simple applications such as TPC-A, however, a TP monitor can be very intrusive. Measurements taken on a VAX 6300 system running the TPC-A benchmark and using ACMS revealed that the TP monitor consumes 20 percent to 25 percent of the cycles on the back-end server.

SQL multi-statement procedures in DEC Rdb version 6.0 address both these performance issues. They reduce the transaction code path by compounding a number of SQL statements in one BEGIN-END procedure that fully adheres to the VAX and AXP procedure calling standards. The BEGIN-END procedure is atomically submitted to the database manager to compile once per database session and to execute as often as the application requires for the duration of that particular session.

When multiple SQL statements are bundled into a single BEGIN-END block, only two network I/O operations are required between the client and the remote server for each database request. Through the DEC Rdb remote facility, which is an integral part of DEC Rdb, client-server applications no longer need the services of a TP monitor. Therefore many of the processing cycles that would have been dedicated to the TP monitor are regained and applied toward processing requests.

The DEC Rdb remote server is an ordinary VMS process that is created upon the first remote request to the database. It has less overhead than the TP monitor. The DEC Rdb remote server remains attached to the database; it communicates with and acts on behalf of its client for the duration of a database session. Figure 8 shows our implementation of the TPC-A client-server application with DEC Rdb version 6.0. With the implementation of multi-statement procedures for the TPC-A transaction, we reduced the code path approximately 20 percent to 25 percent.

Performance Measurement

In this section, we briefly describe a TPC-A transaction and the TPC-A benchmark. We discuss our goals for TPC-A and recount our progress. Finally, we present profiling and benchmark results.

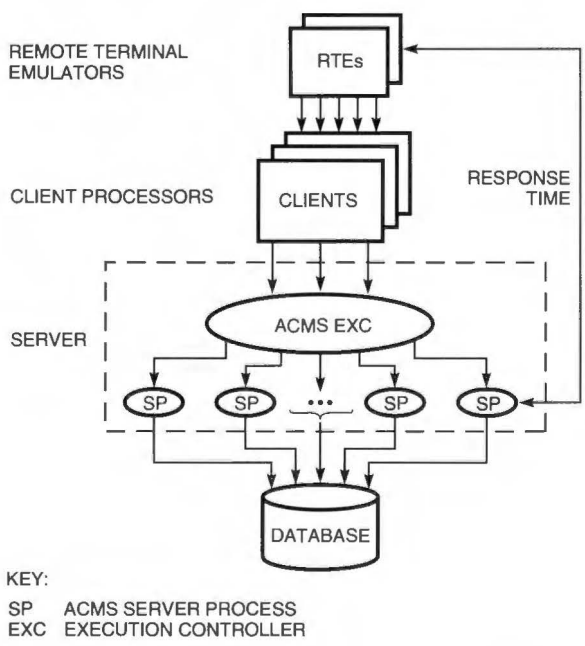


Figure 7 Client-server Application with ACMS

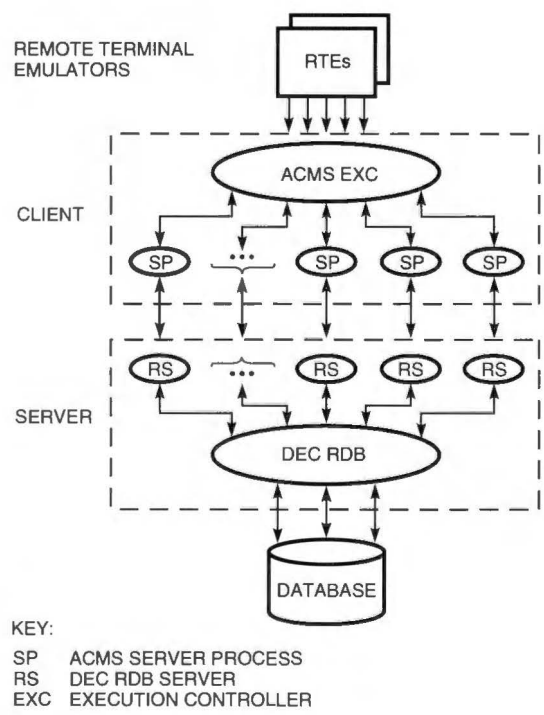


Figure 8 Client-server Application with DEC Rdb Version 6.0

TPC-A Transaction

The TPC-A transaction is a very simple database transaction: a user debits or credits some amount of money from an account. In database terms, that requires four updates within the database: modify the user account balance, modify the branch account balance, modify the teller account balance, and store a history record. The resulting metric indicates how many transactions are performed per second (TPS).[9]

In terms of complexity, the TPC-A transaction falls somewhere in the middle range of benchmarks. In other words, the SPECmark class of benchmarks is very simple and tends to stress processors and caching behavior; the sorting benchmarks (e.g., AlphaSort) expand the scope somewhat to test processors, caches, memory, and I/O capabilities; the TPC-A benchmark tests all the above in addition to stressing the database software (based on the relatively simple transaction). Other benchmarks such as TPC-C and TPC-D place even more emphasis on the database software, thereby overshadowing the hardware ramifications. Hence the TPC-A benchmark is a combined test of a processor and the database software.

During the porting cycle, Digital was aware that the highest result in the industry for a single-processor TPC-A benchmark was approximately 185 TPS. Toward the end of the porting effort, we worked for six to nine months to ensure that DEC Rdb had attained optimal performance for the TPC-A transaction.

In April 1993, the DEC Rdb database system was officially audited on a DEC 10000 AXP system at the world-record rate of 327.99 TPS. As a result, DEC Rdb became the first database system to exceed the 300 TPS mark on a single processor. A few weeks later, DEC Rdb was again audited and achieved 527.73 TPS on a dual-processor DEC 10000 AXP system. Thus, DEC Rdb became the first database system to exceed 500 TPS on a dual-processor machine. Table 3 gives our results; the audits were performed by KPMG Peat Marwick. The mean qualified throughput (MQTh) is the transaction rate at steady state, and $K/tpsA is a measure of the price per transaction for the hardware and software configuration.

Anatomy of the TPC-A Transaction

To understand how DEC Rdb achieves such fast transaction rates, we need to examine the effects of the optimizations to the software with respect to the execution of the TPC-A transaction.

To complete the four updates required by the TPC-A transaction, DEC Rdb actually incurs two physical I/O operations and two lock operations. More specifically, the branch and teller records are located on a page that is cached in the database buffer pool; therefore, these two records are updated extremely fast. The update to the account record causes the account page/record to be fetched from disk and then modified. A history record is then stored on a page that also remains in the buffer pool.

Note that there are too many account records for all of them to be cached in the buffer pool. Hence, the account fetch is the operation that causes pages to cycle through the buffer pool and eventually be flushed to disk. The ABW protocols described previously are utilized to write groups of account pages back to disk asynchronously. (The branch, teller, and history pages never approach the end of the LRU queue.)

In regard to locking, the branch, teller, and history pages/records are all governed by locks that are carried over from one transaction to the next. There is no need to incur any locks to update these records. The account page/record, which must be fetched from disk, requires a new lock request.

At commit time, the after images of the record modifications are submitted to the AIJ log. The new protocols allow the user process to "sleep," while the ALS process flushes the AIJ records to the AIJ file. After the ACE I/O has completed, which occurs before the I/O to the AIJ file on disk, the user processes are "woken up" to begin processing their next transactions.

As shown in Figures 9 and 10, the transaction is dominated by stall times. Since the AXP processors

Table 3 DEC Rdb TPC-A Benchmarks

Processor	Cycle Time	MQTh	$K/tpsA
DEC 7000 AXP Model 610	5.5 ns	302.68	$6,643.00
DEC 7000 AXP Model 610	5.0 ns	327.99	$6,749.00
DEC 7000 AXP Model 620	5.0 ns	527.73	$6,431.00

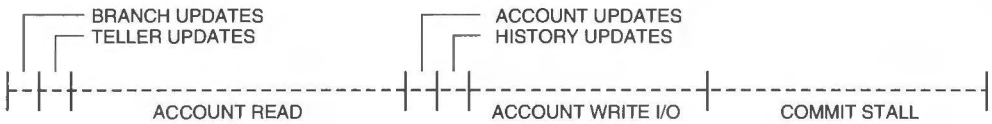


Figure 9 Transaction Duration before Modifications

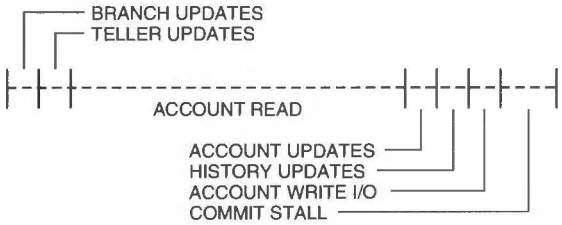


Figure 10 Improved Transaction Duration due to Performance Modifications

are so fast, the branch, teller, and history updates, and the two locks are a very small fraction of the transaction duration. The synchronous account read is a big expense. The batched asynchronous account writes are interesting. Indeed, each transaction requires one account read, which then causes one account write since the buffer pool overflows. Because the account writes are batched into groups and written asynchronously, however, there is no stall time required in the path of the transaction.

Another critical performance metric in the TPC-A benchmark is response time. Figure 11 shows the response times with an average of about 1 second. Figure 12 shows the response time during the steady-state period of the TPC-A experiment.

Performance Profiling of DEC Rdb

In this section, we describe in more detail the instruction profile (i.e., instruction counts, machine cycles, and processor modes) generated during the TPC-A tests.

Performance profiling of DEC Rdb was obtained using Digital's IPROBE tool, the RMU, and the VMS performance monitor. IPROBE is an internal tool built to capture information from the two processor counters that were established to count on-chip events and interrupts after a threshold value was reached.

The TP1 benchmark, a back-end-only version of the TPC-A benchmark, was used to measure DEC Rdb performance on a DEC 7000 AXP Model 610

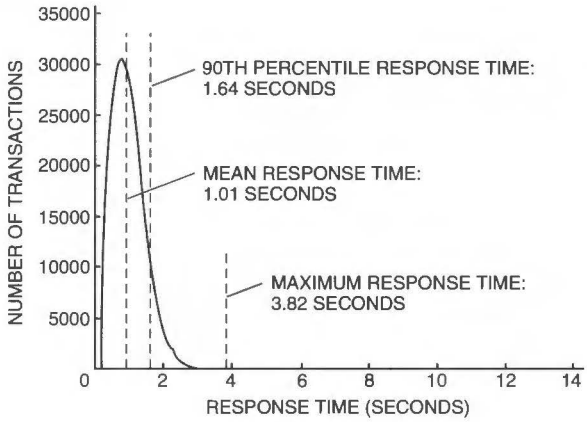


Figure 11 Response Time versus TPS

configured with 5 KDM70 disk controllers and 20 RA70 disk devices. We relied on the IPROBE tool primarily to generate PC (program counter) sampling and to track transaction path length variations as new features were prototyped and performance optimizations were added. We also used IPROBE to verify cache efficiency and the instruction mix in TP applications. We used the RMU to measure the maximum throughput of the TP1 benchmark, the database and application behaviors.

Transaction Cycles Profile

We conducted experiments to determine the performance of TP1 transactions. We used the DEC 7000 AXP Model 610 system (with a 5.5-nanosecond processor) and the OpenVMS AXP version 1.5 operating system. With this configuration, DEC Rdb version 6.0 software achieved a maximum throughput of 334 TPS for the TP1 benchmark. More than 95 percent of the transactions completed in less than 1 second.

Table 4 gives the distribution of the cycles per transaction and the cycles in PALcode in the various CPU modes. The cycle count per TP1 transaction was measured at 544,000 cycles. PALcode calls represent approximately 13 percent of the overall cycle count. In measurements taken with early OpenVMS

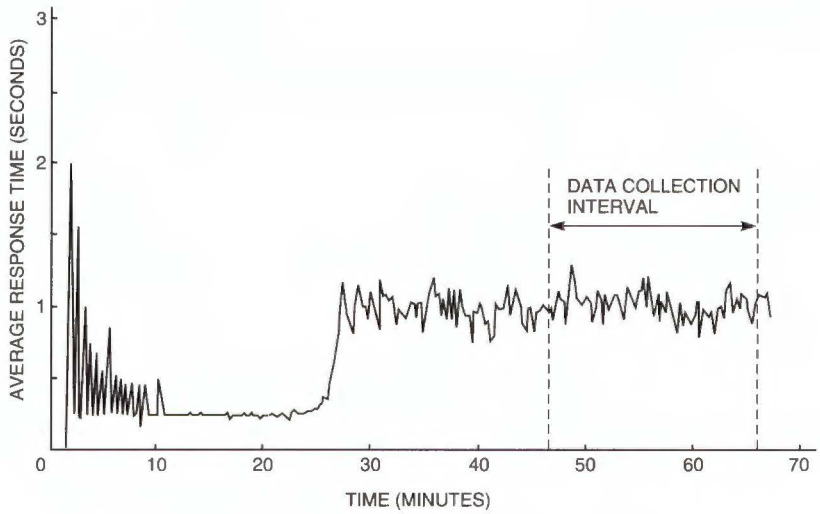


Figure 12 TPC-A Measurement at Steady State

Table 4 Transaction Cycles in CPU Modes

	Cycles per Transaction	% Cycles in System Modes			
		Interrupt	Kernel	Executive	User
Cycles	544.0k	7.1	14.0	73.0	5.8
PALcode cycles	71.5k	13.0	18.0	61.4	7.3
Dual issues	19.0k	5.1	11.6	79.0	4.0

AXP base levels, PALcode calls represented 28 percent of the overall cycle count. The most frequently called PALcode functions were misses in the data and instruction translation buffers. Four major DEC Rdb images may now be installed "/RESIDENT" to take advantage of granularity hints and reduce misses in the instruction translation buffer. Dual issuing remained low throughout the experiments we conducted with various DEC Rdb and OpenVMS AXP base levels.

TP1 Transaction Path Length Profile

At the initial stage of the DEC Rdb port to the Alpha AXP platform in January 1993, the TP1 transaction path length was measured at 300,000 reduced instruction set computer (RISC) instructions with the available OpenVMS AXP base levels. In April 1993, the TP1 transaction path length dropped to 189,000 RISC instructions. As shown in Table 5, the TP1 transaction path length is currently at 133,500 RISC instructions. That measurement is 30 percent less than it was in April 1993. The cycles per instruction were measured at 3.8.

Summary

Based on current performance and future trends, the Alpha AXP family of processors and platforms will provide superb servers for high-end production

Table 5 TP1 Transaction Path Length

MQTh	Transaction Path Length	Cycles per Instruction	% Time in CPU Modes			
			Interrupt	Kernel	Executive	User
334 TP1 TPS	133.5k	3.8	4.8	11.3	79.9	3.4

systems. To keep pace with the phenomenal increases in CPU speed, database systems must incorporate features that reduce the I/O bottleneck. DEC Rdb version 6.0 software has incorporated a number of these features: asynchronous page fetch, asynchronous batch write, multithreaded backup and restore, multi-statement procedures, and AIJ log server using electronic caches. These enhancements not only allow optimal transaction processing performance but also permit systems to deal with very large data sets.

Acknowledgments

The development of DEC Rdb V6.0 was a team effort involving more people than can be acknowledged here. We would, however, like to recognize the significant contributions of Jay Feenan, Richard Pledereder, and Scott Matsumoto for their design of the multi-statement procedures feature of DEC Rdb and of Ed Fisher for his work on the Rdb code generator.

References

1. L. Hobbs and K. England, *Rdb/VMS: A Comprehensive Guide* (Burlington, MA: Digital Press, 1991).
2. R. Sites, "Alpha AXP Architecture," *Digital Technical Journal*, vol. 4, no. 4 (1992): 19–34.
3. J. Ousterhout and F. Douglas, "Beating the I/O Bottleneck: A Case for Log-Structured File Systems," Technical Report, University of California at Berkeley (1988).
4. J. Coffler, Z. Mohamed, and P. Spiro, "Porting Digital's Database Management Products to the Alpha AXP Platform," *Digital Technical Journal*, vol. 4, no. 4 (1992): 153–164.
5. D. Lomet, R. Anderson, T. Rengarajan, and P. Spiro, "How the Rdb/VMS Data Sharing System Became Fast," Technical Report CRL 92/4, Digital Equipment Corporation, Cambridge Research Laboratory (1992).
6. P. Spiro, A. Joshi, and T. Rengarajan, "Designing an Optimized Transaction Commit Protocol," *Digital Technical Journal*, vol. 3, no. 1 (Winter 1991): 70–78.
7. E. Knuth, *Sorting and Searching, The Art of Computer Programming* (Reading, MA: Addison-Wesley Publishing Company, 1973).
8. C. Nyberg, T. Barclay, Z. Cvetanovic, J. Gray, and D. Lomet, "AlphaSort: A RISC Machine Sort," Technical Report 93.2, Digital Equipment Corporation, San Francisco Systems Center (1993).
9. J. Gray, *The Benchmark Handbook* (San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1993).

William L. Goleman
Robert G. Thomson
Paul J. Houlihan

Improving Process to Increase Productivity While Assuring Quality: A Case Study of the Volume Shadowing Port to OpenVMS AXP

The volume shadowing team achieved a high-quality, accelerated delivery of volume shadowing on OpenVMS AXP by applying techniques from academic and industry literature to Digital's commercial setting. These techniques were an assessment of the team process to identify deficiencies, formal inspections to detect most porting defects before testing, and principles of experimental design in the testing to efficiently isolate defects and assure quality. This paper describes how a small team can adopt new practices and improve product quality independent of the larger organization and demonstrates how this led to a more enjoyable, productive, and predictable work environment.

To achieve VMScluster support in the OpenVMS AXP version 1.5 operating system one year ahead of the original plan, OpenVMS Engineering had to forego early support of Volume Shadowing Phase II (or "shadowing"). Shadowing is an OpenVMS system-integrated product that transparently replicates data on one or more disk storage devices. A shadow set is composed of all the disks that are shadowing (or mirroring) a given set of data. Each disk in a shadow set is referred to as a shadow set member. Should a failure occur in the software, hardware, firmware, or storage media associated with one member of a shadow set, shadowing can access the data from another member.

The ability to survive storage failures is quite important to customers of OpenVMS systems where data loss or inaccessibility is extremely costly. Such customers typically combine shadowing and VMScluster technologies to eliminate single points of failure and thereby increase data availability. For these customers, delayed support for shadowing on the OpenVMS AXP system meant either foregoing the advanced capabilities of an Alpha AXP processor within their VMScluster systems or foregoing the additional data availability that shadowing provides. To resolve this dilemma, OpenVMS Engineering began a separate project to rapidly port shadowing to the OpenVMS AXP system. This project had three overall goals.

- Provide performance and functionality equivalent to the OpenVMS VAX system
- Allow trouble-free interoperability across a mixed-architecture VMScluster system
- Deliver to customers at the earliest possible date

All three goals were met with the separate release of shadowing based on OpenVMS AXP version 1.5 in November 1993, more than six months ahead of the original planned release for this support.

In the following sections, we describe how we achieved these goals by reshaping our overall process, reworking our development framework, and redirecting our testing. In the final section on project results, we demonstrate how our improved process assures quality and increases productivity. This paper assumes familiarity with the shadowing product and terminology, which are described fully in other publications.[1,2]

Reshaping the Overall Process

Because the need was urgent and the project well-defined, we could have leapt directly into porting the shadowing code. Instead, we took a step back to evaluate how best to deliver the required functionality in the shortest time and how best to verify success. Doing so meant taking control of our software development process.

Effective software process is generally acknowledged as essential to delivering quality software products. The Capability Maturity Model (CMM) developed by the Software Engineering Institute embodies this viewpoint and suggests that evolving an entire organization's process takes time.[3,4] Grady and Caswell's experience implementing a metrics program at Hewlett-Packard bears out this viewpoint.[5] Our experience with the continuous improvement of software development practices within Digital's OpenVMS Engineering does so as well.

However, our engineering experience also suggests that the current emphasis on evolving an entire organization's process tends to overshadow the ability of a small group to accelerate the adoption of better engineering practices. Within the context of an individual software project, we believed that process could be readily reshaped and enhanced in response to specific project challenges. We further believed that such enhancements could significantly improve project productivity and predictability.

Identifying Process Challenges

At the project's outset, we identified four major challenges that we believed the project faced: configuration complexity, defect isolation costs, beta test ineffectiveness, and resource constraints.

Configuration Complexity Our most significant challenge was to devise a process to efficiently validate the product's complex operating environment: a mixed-architecture VMScluster system comprising both Alpha AXP and VAX processors (or nodes).[6] Digital's VMScluster technology currently supports a configuration of loosely coupled, distributed systems comprising as many as 96 AXP and VAX processors. These nodes may communicate over any combination of four different system interconnects: Computer Interconnect (CI), Digital Storage Systems Interconnect (DSSI), fiber distributed data interface (FDDI), and Ethernet. VMScluster systems support two disk storage architectures—the Digital Storage Architecture (DSA) and the small computer systems interface (SCSI)—and dozens of disk models. Once ported, shadowing would be required to provide a consistent view across all nodes of as many as 130 shadow sets. Each shadow set may involve a different model of disk and may span different controllers, interconnects, nodes, or processor architectures. The potential number of configuration variations is exponential.

Defect Isolation Costs A second major process challenge was to contain the cost of isolating defects. A defect is defined to be the underlying flaw in the OpenVMS software that prevents a VMScluster system from meeting customer needs. System software defects can be triggered by VMScluster hardware, firmware, and software. Since few individuals possess the combined skills necessary to troubleshoot all three areas, defect isolation normally involves a team of professionals, which adds to the cost of troubleshooting VMScluster operating system software.

Debugging of shadowing code is difficult since it executes in the restricted OpenVMS driver environment: in kernel mode at elevated interrupt priority level. Shadowing is also written mostly in assembly language. To maintain shadow set consistency across all 96 nodes of a VMScluster system, much of the shadowing code involves distributed algorithms. Troubleshooting distributed algorithms can greatly increase isolation costs, since a given node failure is often only incidental to a hardware, firmware, or software defect occurring earlier on another VMScluster node.

Many shadowing problem reports ultimately prove to contain insufficient data for isolating the problem. Other problem reports describe user errors or hardware problems; some are duplicates. For example, Figure 1 shows the trend for Volume Shadowing Phase II problems reported, problems resolved, and defects removed between December 1992 and April 1993. During this period, only one

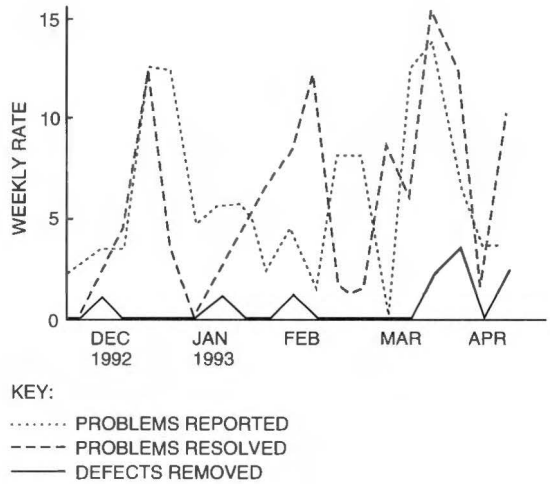


Figure 1 Problem Handling and Defect Removal on VAX: December 1992 to April 1993

defect was fixed for every ten problem reports closed. Because this low ratio is not typical of most OpenVMS subsystems, it is not readily accommodated by our traditional development process.

Beta Test Ineffectiveness A third process challenge was that customer beta testing had not contributed significantly to shadowing defect detection. Justifiably, most customers simply cannot risk incorporating beta test software into the kind of complex production systems that are most likely to uncover shadowing problems. Figure 2 shows the distribution of shadowing problem reports received from its inception in January 1990 to January 1993. During these three years, only 8 percent of the problem reports came from customer beta test sites. In contrast, 46 percent of the problem reports came from stress test and alpha test sites within Digital, where testing was based on large, complex VMScluster configurations.

Resource Constraints A fourth process challenge for the shadowing port was competition for engineering resources. Only the development and validation project leaders could be assigned full-time. The ongoing demands of supporting shadowing on OpenVMS VAX precluded members of the existing shadowing team from participating in the port. Most other engineering resources were already committed to the accelerated delivery of VMScluster support in OpenVMS AXP version 1.5. As a consequence, the majority of the shadowing team comprised experienced OpenVMS engineers whose familiarity with shadowing was limited, whose individual skill sets were often incomplete for this particular project, and whose availability

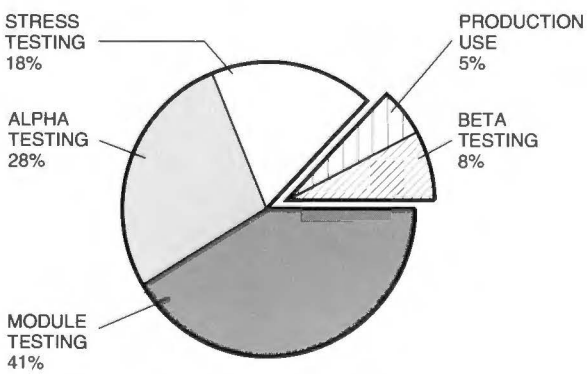


Figure 2 Sources of Shadowing Problem Reports: January 1990 through January 1993

was staggered over the course of the project. Moreover, the team was split between the United States and Scotland and, hence, separated by a six-hour time difference.

Making Process Enhancements

To meet these challenges, we believed our overall process required enhancements that would provide

- Independent porting tasks within a collaborative and unifying development framework
- Aggressive defect removal with an emphasis on containing porting defects
- Directed system testing that preceded large-scale stress testing
- Clear validation of shadowing's basic error-handling capabilities

Figure 3 shows our reshaped process for the shadowing port. Each step in the process is depicted in a square box starting with planning and ending with the project completion review. New steps in the process are shaded gray. The most significant enhancements were the insertion of inspection and profile testing steps. To evaluate our progress in removing defects, we incorporated defect projections for each development step into our release criteria. To track this progress, we supplemented the organization's problem-reporting database with a project-defect database. Emphasizing error insertion during profile and acceptance test allowed for validation of shadowing's error-handling capabilities.

In making these process enhancements, we were careful to maintain both consistency with prevailing industry practices and compatibility with current practices within OpenVMS Engineering. We felt that adopting ideas proven in industry and having a common framework for communication within our organization would increase the probability of success for our enhancements. How we implemented these enhancements is described in the following sections.

Measuring Process Effectiveness

Establishing Release Criteria In formulating the release criteria for shadowing given in Table 1, we used Perry's approach of

- Establishing the quality factors that are important to the product's success
- Mapping the factors onto a set of corresponding attributes that the software must exhibit

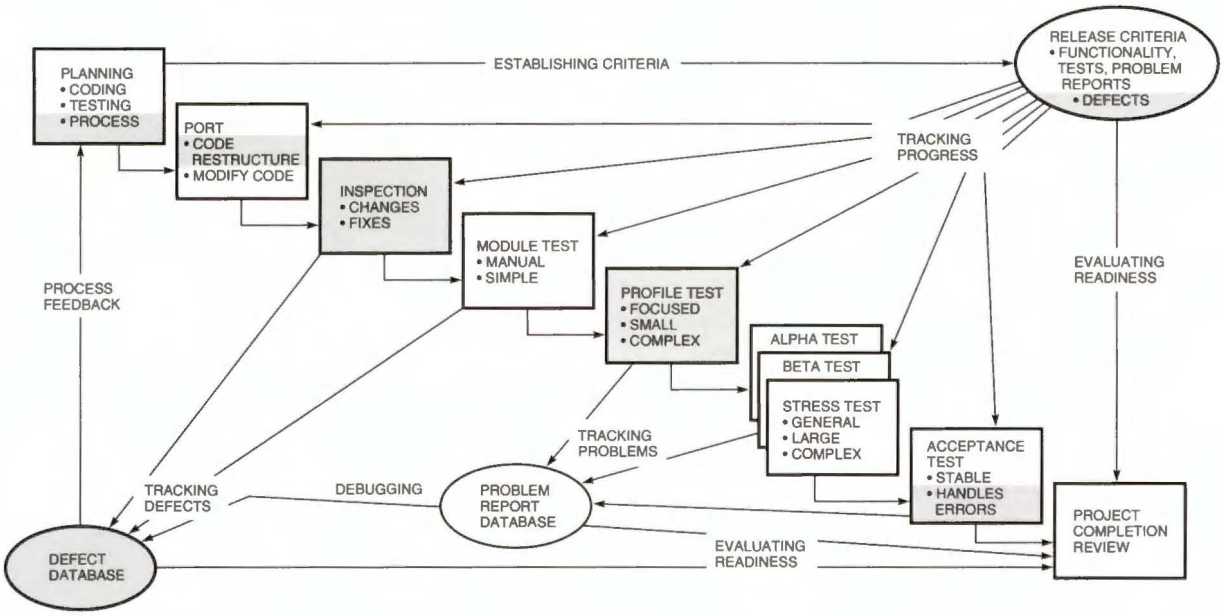


Figure 3 Enhanced Development and Validation Process

Table 1 Shadowing Release Criteria

Quality Factor	Software Attribute	Software Metric	Threshold Value
Reliability	Error tolerance	Profile test completion	100%
	Operational accuracy	Defects detected	240
	Operational consistency	Incoming problem reports per week	Near 0
		Acceptance test with error insertion	200 node-hours
Integrity	Data security	Unresolved high-severity defects	None
Correctness	Completeness	Code ported	100%
		Module test completion	100%
		Code change rate per week	0
Efficiency	Processing time	Queue I/O reads and writes	Comparable to VAX
	Throughput	Copy and merge operations	
Usability	Ease of training	Documentation completion	100%
Interoperability	Backward compatibility	Stress test completion	12,000 node-hours
	Transparent recovery	Unresolved high-severity defects	None
Maintainability	Self-descriptiveness	Source modules restructured	100%
	Consistency		

- Identifying metrics and threshold values for determining when these software attributes are present[7]

Defining release criteria based on threshold values provided a clear standard for judging release readiness independent of the project schedule. These criteria spanned the development cycle in order to provide a basis for verifying progress at each stage of the project. The emphasis of most metrics for these criteria was on containing and removing defects. Other metrics were selected to corroborate that high defect detection equated to high product quality.

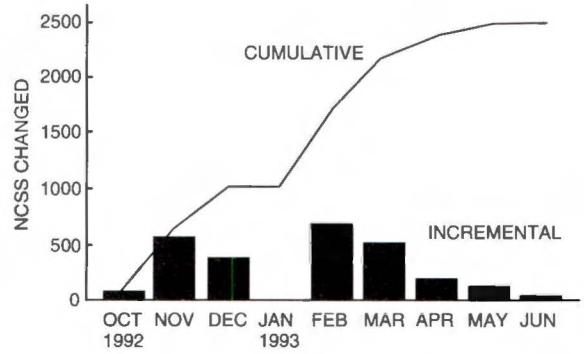


Figure 4 Projections of Code Changed by Month

Tracking Defects Projecting defect detection levels for all stages in the development cycle was a departure from the traditional practice of our development engineers. Previously, only the test engineers within OpenVMS Engineering established a defect goal prior to beginning their project work. Extending the scope of this goal for shadowing resulted in a paradigm shift that permeated the entire team's thinking and encouraged each member to aggressively look for defects. Since all team members were more committed to meeting or exceeding the defect goal, they were eager to provide detailed information on the circumstances surrounding defect detection and removal. This detail is often lost in a traditional development project.

Because data on code modification and defect removal during an OpenVMS port was not readily available, we derived our projections as follows.

1. We determined how many lines of code would be modified during the shadowing port. This estimate was based on a comparison between the ported and original sources for another OpenVMS component of similar complexity. The resulting estimate for shadowing changes was 2,500 noncomment source statements (NCSS) out of a total of roughly 19,400.

2. We projected the rate at which these modifications would occur for shadowing. We based this projection on the actual time spent porting, inspecting, and debugging the code of a second OpenVMS component of similar complexity. We revised it upon reaching the first major porting milestone to reflect our actual performance. The revised projection is shown by month in Figure 4.

3. Using the results from our first milestone, we estimated that 250 defects would be introduced as a result of the complete port. This estimate included not only defects introduced through code modifications but also defects induced in existing code by these modifications.

4. We projected the schedule of defect detection for each stage of the development cycle. This projection assumed that defects were distributed uniformly throughout the code. Based again on the results of our first porting milestone, we estimated that our efficiency at removing the defects in this release (or defect yield) would be roughly 60 percent through inspections and 25 percent through module testing. We assumed that an additional 10 percent of the defects would be removed during profile and stress testing. A 95 percent overall yield for the release is consistent with the historic data shown in Figure 2. It is also consistent with the highest levels of defect removal efficiency observed in the industry where formal code inspections, quality assurance, and formal testing are practiced.[8] Figure 5 shows our projections for removing 240 defects (95 percent yield of our estimate of 250 defects) by both month and method.

Tracking defects was difficult within our larger organization because the problem reporting system used by OpenVMS Engineering did not distinguish between defects, problem reports, and general communication with test sites. To work around this shortcoming, we created a project database to track defects using off-the-shelf personal computer (PC) software to link defects to problem reports.

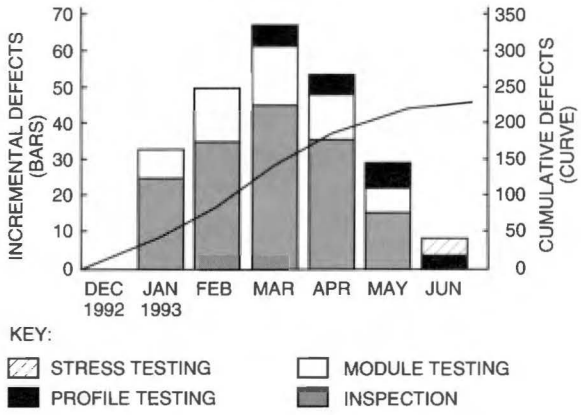


Figure 5 Projected Defect Detection by Month and Method

Reworking the Development Framework

Only the development project leader satisfied all the requirements for executing a rapid port of the shadowing code:

- Expertise in shadowing, VMScluster systems, OpenVMS drivers, the VAX assembly language, the AMACRO compiler (which compiles VAX assembly language for execution on Alpha AXP systems), and the Alpha AXP architecture[9]
- Experience porting OpenVMS code from the VAX to the Alpha AXP platform[9]
- Familiarity with the defect history of shadowing and the fixes that were being concurrently applied to the VAX shadowing code
- Availability throughout the duration of the project to work on porting tasks at the same time and the same place

To compensate for the lack of these capabilities across all team members and to improve our ability to efficiently port the code while minimizing the number of defects introduced, we reworked our development framework in two ways. First, we restructured the modules to improve their portability and maintainability. Second, we inspected all porting changes and most fixes to assure uniform quality across both the project and the code.

Restructuring Modules

Examining the interconnections between the shadowing modules in the OpenVMS VAX software revealed a high degree of interdependence based on content coupling.[10] Modules frequently branched between one another with one module using data or control information maintained in another module. These modules also exhibited low cohesion with subroutines grouped somewhat by logical ordering but primarily by convenience.[10]

Structured in this fashion, the shadowing modules were not only more difficult to maintain but also less separable into independent porting tasks. Moreover, differences between the VAX and the Alpha AXP architectures, together with the transformation of the VAX assembly language from a machine assembly language to a compiled language, precluded the continued use of content coupling in the ported code.

To remedy these structural problems, we partitioned the shadowing code into functional pieces that could be ported, inspected, and module tested separately before being reintegrated for profile and stress testing. Figure 6 shows both the original and the reworked relationships between shadowing's source modules and its functions. During restructuring, we emphasized not only greater functional cohesion within the modules but also improved coupling based primarily on global data areas and I/O interfaces. As a consequence, most shadowing functions were directly dependent on only one other function: mounting a single member. Once the port of this function was complete, all the others could be largely ported in parallel. Where a particular module was used by more than one function, we coordinated our porting work using a scheme for marking the code to indicate portions that had not been ported or tested.

Inspecting Changes

We believed inspections would serve well as a tool for containing our porting defects. Industry literature is replete with data on the effectiveness of inspection as well as guidelines for its use.[8,11,12] Since our code was now structured for parallel porting activities, however, the project also needed a framework for

- Integrating engineers into the project
- Coordinating overlapping tasks
- Collaborating on technical problems
- Sharing technical expertise and insights
- Assuring that the engineers who would maintain shadowing understood all porting changes

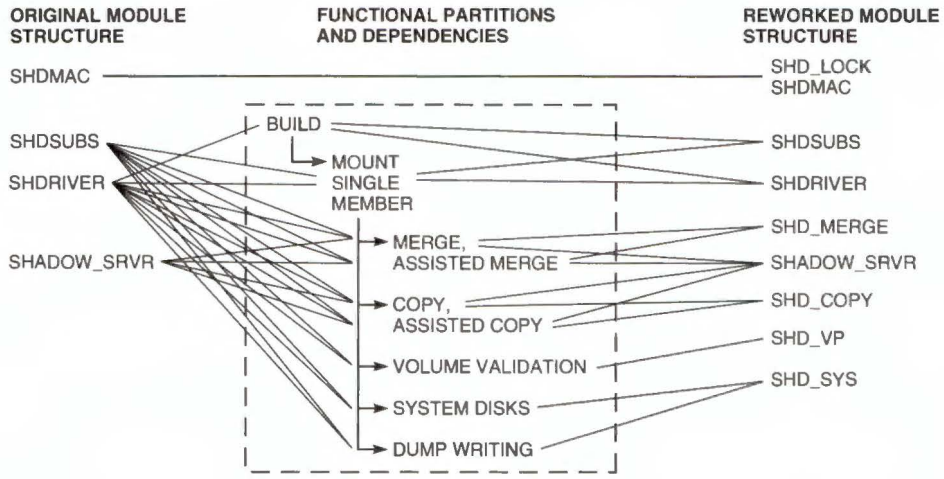


Figure 6 Code Restructuring by Shadowing Function

We believed that group inspections could provide this framework.

Tailoring the Inspection Process Our inspections differed from the typical processes used for new code.[12] Only the changes required to enable the VAX code to execute correctly on the Alpha AXP platform were made during the port. These changes were scattered throughout the sources, primarily at subroutine entry points. With our time and engineering resources quite constrained, we chose to inspect only these changes and not the entire code base. Because the project involved the port of an existing, stable product, no new functionality was being introduced and therefore no functional or design specifications were available. Instead, inspections were made using the VAX code sources as a reference document.

Integrating Engineering Resources Prior experience in OpenVMS Engineering indicated that engineers working on unfamiliar software could be very productive if they worked from a detailed specification and used inspections. Using the VAX sources as a "specification" for the shadowing port provided such a focus. Inspecting only code changes alleviated the need for team members to understand how a particular shadowing function worked in its entirety. Simply verifying that the algorithm in the ported sources was the same as that in the original VAX sources was sufficient.

Inspections of ported code preceded both module testing and the integration of the porting changes into the overall shadowing code base. This assured that any differences in coding standards or conventions were harmonized and any misunderstanding of code operation was corrected before the code underwent module testing. Inspections occurred before the engineers who performed the port returned to their original duties within OpenVMS Engineering. By participating in these inspections, the engineers who would maintain the ported code understood exactly what was changed, in case additional debug work was needed.

Sharing Experience and Expertise The inspection process provided a forum for team members to share technical tips, folklore, background, and experience. Having such a forum enabled the entire team to leverage the diverse technical expertise of its individual members. The resulting technical synergy increased the capacity of the team to execute the porting work. It also led to rapid cross-training between team members so that everyone's technical skills increased during the course of the project. This teamwork and increased productivity led to more enjoyable work for the engineers involved.

In retrospect, the use of inspections proved the greatest single factor in enabling the project to meet its aggressive delivery schedule.

Redirecting the Testing

To detect both new and existing defects, shadowing has historically undergone limited functional testing followed by extensive stress testing. The

effectiveness of this testing has been constrained, however, because it

- Provided no measurement of actual code coverage
- Lacked an automatic means for forcing the execution of error paths within shadowing
- Failed to target the scenarios in which most shadowing failures occurred

To compensate for these shortcomings and improve our ability to efficiently detect defects, we formulated profile testing: a method of risk-directed testing that would follow module testing and precede large-scale stress testing.

Defining Profile Testing

Profile testing focuses on operating scenarios that pose the greatest risk to a software product. Engineering experience clearly indicated that the highest-risk operating scenarios for shadowing involved error handling during error recovery. Examples of such scenarios include media failure after a node failure or the unavailability of system memory while handling media failure. Problems with such error handling have typically occurred only in large and complex VMScluster systems. Test profiles are simple, clearly defined loads and configurations designed to simulate the complex error scenarios and large configurations traditionally needed to detect shadowing defects.

Fundamentally, profile testing is the application of the principles of experimental design to the challenge of "searching" for defects in a large test domain. Deriving profile tests begins with a careful identification of operating conditions in which the product has the greatest risk of failing. These conditions are then reduced to a set of hardware and software variables (or "factors") and a range of values for these factors. A test profile is the unique combination of factor values used in a test run.

Combining a large set of test factors and factor values can result in unmanageable complexity. For this reason, profile testing uses orthogonal arrays to select factor combinations for testing. These arrays guarantee uniform coverage of the target test domain described by the test factors. Instead of selecting tests based on an engineer's ingenuity, we used these arrays to systematically select a subset of all possible factor combinations. As a result, we uncovered nonobvious situations that customers often encounter. By relying on combinatorics rather than randomness to detect defects, the event sequences leading to a defect can be more readily reproduced.

The following sections show how we used this approach to design and implement profile testing for shadowing. They also reveal that the cost-effectiveness of our testing improved significantly as a result.

Describing the Test Domain

Both AT&T and Hewlett-Packard have used operational profiles to describe the test domain of complex software systems.[13] Such operational profiles were typically derived by monitoring the customer usage of the system and then methodically reducing this usage to a set of frequencies for the occurrence of various systems functions. These frequencies are then used to prioritize system testing. For the purposes of validating shadowing, we extended this notion of an operational profile by decomposing the test domain into four distinct dimensions that characterize complex software systems:

- System configuration
- Software resources
- Operational sequences
- Error events

The emphasis of our test profiles was not on how the system was likely to operate, but rather on how it was likely to fail. For our project, rapid characterization of the test domain was of greater importance than precise reproduction of it.

Identifying the Test Factors

Assessment of shadowing's test domain identified the factors that characterized its high-risk operating scenarios. This assessment was based on a review by both test and development engineers of the product's functional complexity, defect history, and code structure as characterized by its cyclomatic complexity.[14] The resulting factors provided the basis for formulating our test profiles.

System Configuration The following key factors in system configuration describe how a shadow set is formed and accessed across the range of components and interconnects that VMScluster systems support.[6]

- Number of shadow set members (MEMBCNT)
- Device MSCP serving (MSCPSERV)
- Controller sharing (SEPCTRL)
- Emulated versus local disk controller (SERVSYS)
- Alpha AXP or VAX I/O load initiator (LOADSYS)
- Location of the storage control block (SCBBEG)
- Size limits for I/O transfers (DIFMBYT)
- Controller time-out values (DIFCTMO)
- System disk shadow set (SYSDISK)
- Disk device type (DISKTYPE)

Software Resources Although running an application in OpenVMS can involve competing for a wide range of finite system and process resources, only two software resources initially appeared significant for targeting error handling during error recovery within the shadowing product:

- System memory used for I/O operations
- VMScluster communication resource (send credits)

Operational Sequences Shadow set membership is controlled by the manager of an OpenVMS system.[2] The manager initially forms the shadow set and then adds or removes members as needed. Applications use these shadow sets for file creation and access. During its use, a shadow set can require copy and merge operations to maintain data consistency and correctness across its members. Profiles that target these activities involve sequences of the following key operations.

- Merge, assisted merge, copy, and assisted copy
- Member mounts and dismounts
- File creation and deletion
- Random reads and writes; repeated reads of a "hot" block on the disk

Error Events All complex software systems must deal with error events that destabilize its operation. For the shadowing product, however, reliably handling the following set of errors represents the essence of its value to OpenVMS customers.

- Removal of a VMScluster node (NODEERR)
- Process cancellation (PROCERR)
- Controller failure (CTRLERR)
- Disk failure (DISKERR)
- Media failure (MEDIAERR)

Managing Test Complexity

When the list of key factors for targeting shadowing's high-risk operating scenarios was enumerated, the resulting test domain was unmanageably complex. If just two test values for each of 25 factors are assumed, the set of all possible combinations was 2^{25} or more than 33 million test cases.

To reduce this combinatorial complexity to a manageable level, we structured profiles in two test dimensions, error event and system configuration, using orthogonal arrays.[15,16] Columns in these orthogonal arrays describe the test factors, and rows describe a balanced and orthogonal fraction of the full set of factor combinations. Because of their balance and orthogonality across the test factors, such arrays provide a uniform coverage of the test domain.

Figure 7 is a composite table of error-event profiles created using a D-optimal array and shadow set configuration profiles created using a standard array.[17,18] These two arrays formed the base of our profile test design. Operational sequences were applied (as described below) to 18 test profiles formed by relating the error event and shadow set arrays as shown in Figure 7. These profiles were arranged into groups of three; each group included two types of disks, a shadowed system disk, and the presence of each type of disk error. The test values assigned to the factors SYSDISK and DISKTYPE as a result of this grouping are shown in the two columns positioned between the arrays in Figure 7.

Note that physically configuring the test environment prevented us, in some instances, from using the prescribed assignment of factor values. As a consequence, only those factors whose columns are shaded in gray in Figure 7 retained their balance and orthogonality during implementation of the test design.

At this point, profile test design is complete. The use of orthogonal arrays allowed us to reduce the tests to a manageable number and at the same time have uniform coverage of all test factors.

Configuring the Test Environment

In addition to the shadow set profiles, the following practical constraints guided the configuration of a VMScluster system for conducting our profile testing.

PROFILES	NODEERR	PROCERR	CTRLERR	DISKERR	MEDIAERR	SYSDISK	DISKTYPE	MEMBCNT	MSCPSERV	SEPCTRL	SERVSYS	LOADSYS	SCBBEG	DIFMBYT	DIFCTMO
1	NO	NO	NO	FATAL	NO	NO	RZ	2	SOME	NONE	AXP	AXP	ALL	NONE	NONE
2	NO	NO	YES	MNTVER	YES	NO	RF	2	SOME	ALL	BOTH	AXP	ALL	ALL	ALL
3	NO	YES	NO	NO	NO	YES	RF	2	NONE	ALL	BOTH	AXP	NONE	NONE	NONE
4	YES	NO	NO	NO	YES	NO	RZ	2	ALL	ALL	AXP	AXP	ALL	NONE	NONE
5	NO	NO	YES	MNTVER	NO	YES	RF	2	ALL	SOME	BOTH	AXP	NONE	NONE	NONE
6	NO	YES	YES	FATAL	YES	NO	RZ	2	ALL	NONE	VAX	AXP	NONE	NONE	NONE
7	NO	YES	NO	MNTVER	YES	NO	RA	2	NONE	NONE	BOTH	VAX	NONE	NONE	NONE
8	YES	NO	NO	FATAL	NO	NO	RZ	2	NONE	ALL	BOTH	BOTH	ALL	NONE	NONE
9	NO	YES	YES	NO	NO	YES	RZ	2	ALL	NONE	AXP	AXP	NONE	NONE	NONE
10	YES	YES	NO	MNTVER	YES	NO	RZ	3	NONE	SOME	VAX	VAX	NONE	NONE	NONE
11	NO	NO	NO	FATAL	NO	YES	RA	3	NONE	NONE	AXP	AXP	ALL	NONE	NONE
12	YES	YES	YES	NO	NO	NO	RA	3	NONE	SOME	AXP	BOTH	NONE	NONE	SOME
13	YES	YES	NO	MNTVER	NO	NO	RZ	3	ALL	ALL	BOTH	VAX	NONE	SOME	SOME
14	YES	YES	YES	FATAL	YES	YES	RA	3	ALL	NONE	VAX	AXP	NONE	NONE	NONE
15	YES	NO	YES	NO	YES	NO	RZ	3	ALL	NONE	BOTH	AXP	ALL	NONE	NONE
16	YES	NO	YES	MNTVER	NO	NO	RZ	3	SOME	ALL	BOTH	VAX	ALL	SOME	SOME
17	NO	NO	NO	NO	YES	YES	RF	3	SOME	ALL	AXP	AXP	NONE	SOME	SOME
18	YES	YES	YES	FATAL	YES	NO	RZ	3	SOME	SOME	BOTH	BOTH	ALL	SOME	SOME

ERROR EVENT ARRAY L18 ($2^4 \times 3^1$) SHADOW SET CONFIGURATION ARRAY L18 ($2^1 \times 3^7$)

Figure 7 Set of Composite Test Profiles

- Minimize hardware requirements and maximize ease of test execution
- Configure profiles from the shadow set array in groups of three to expedite test execution
- Reflect both anticipated customer usage of shadowing and historic usage as characterized in existing surveys of VAXcluster sites in the United States
- Enable the formation of either one integrated or two separate VMScluster systems based on either the DSSI or the CI system interconnect
- Require no physical reconfiguration during testing
- Maintain a consistent batch/print and user authorization environment
- Follow the configuration guidelines set forth in Digital's software product descriptions for OpenVMS, VMScluster systems, and volume shadowing for Alpha AXP and VAX systems

Constructing a test configuration that reflected all these constraints and supported the shadow set profiles in Figure 7 was quite a challenge. As a result of having clear configuration guidelines, however, we could re-create shadowing's high-risk operating scenarios using substantially less hardware than required for large-scale stress testing. Table 2 contrasts the hardware requirements of the two test approaches.

Table 2 VMScluster Configuration Size by Test Method

	Profile Testing	Large-scale Stress Testing
Systems	5 AXP 4 VAX	10 AXP 12 VAX
Interconnects	1 CI 1 Ethernet	2 CI 9 Ethernet 1 FDDI
CI Storage Controllers	1 HSC 1 HSJ	6 HSC 1 HSJ
Test Disks	31	165
Shadow Sets	9 two-member 9 three-member	23 two-member 5 three-member

The resulting test configuration for our profile testing was formally described in a configuration diagram, which appears in a simplified form in Figure 8. During our testing, each profile shown in Figure 7 was uniquely marked on this diagram to show both the disks comprising each shadow set and the nodes to load them. Taken together, Figures 7 and 8 proved quite useful in transferring the task of executing a particular test profile from one test engineer to another. In addition, development engineers found them to be invaluable tools for clarifying the fault loads and the configuration of a particular test profile.

To illustrate how we used these two tools, consider the first three test profiles shown in Figure 7.

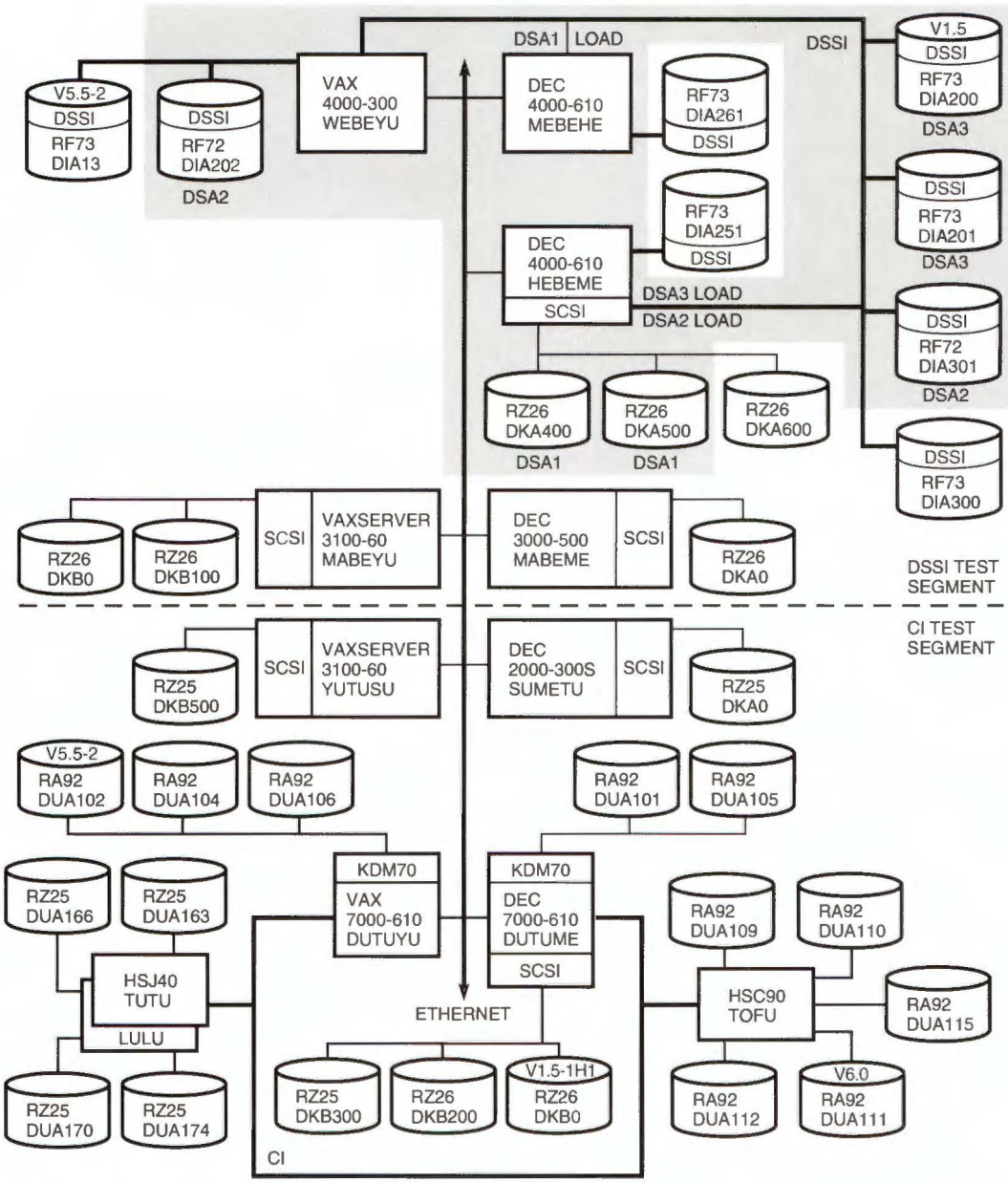


Figure 8 Total System Configuration for Profile Testing

These profiles, respectively, define the configuration and loading of shadow sets DSA1, DSA2, and DSA3. Running these three profiles in parallel would be difficult to describe precisely without these two tools. As an example, profile 1 in Figure 7 indicates that DSA1 must comprise two RZ devices that are not used as system disks. At least one of these devices must be MSCP served. The two disks must share a controller and must be accessed via an AXP node. Both must have their storage control blocks located at their first logical block. As a result of their physical configuration, both must have the same size limit on I/O transfers and the same controller time-out value. Finally, this profile indicates that another AXP node in the VMScluster system must load DSA1 and that this load must involve the simulation of fatal disk errors. Devices DKA400 and DKA500 in Figure 8 satisfied these requirements; the load was to be applied from node MEBEHE.

The complete configuration for these three profiles is denoted by the gray box in Figure 8, which requires only a small subset of the total test configuration to execute

- Two AXP systems (MEBEHE and HEBEME)
- One VAX system (WEBEYU)
- One DSSI interconnect
- One SCSI and 4 DSSI controllers
- Two RZ26 disks (DKA400 and DKA500)
- Two RF72 disks (DIA301 and DIA202)
- Two RF73 disks (DIA200 and DIA201)

Executing Test Profiles

Testing Tools Executing the profile tests involved the use of four Digital internal test tools (XQPXR, IOX, CTM, Faulty Towers) and two OpenVMS utilities (BACKUP and MONITOR). XQPXR and IOX both provided read and/or write loads to shadow sets with XQPXR utilizing the file system for its I/O. CTM provided a means of loading multiple subsystems across the VMScluster system. Faulty Towers was used to inject faults into the VMScluster System Communication Architecture (SCA) protocol tower during loading to create the error profiles shown in Figure 7. MONITOR measured the loads applied during profile testing. BACKUP was used to verify that the data on shadow set members was consistent following a test run.

Of all the test tools we used, Faulty Towers was both the most critical to our success and the most innovative in simulating large-scale VMScluster environments. Historically, large-scale stress testing of shadowing has depended largely on the occurrence of random events or manual intervention to exercise shadowing error paths. Because SCA underlies all communication within a VMScluster system, Faulty Towers could instead automatically force the exercise of these paths by simulating errors within the system. The set of faults that Faulty Towers provided came from our examination of how VMScluster systems, especially large-scale systems, fail. This set included forcing esoteric states throughout the VMScluster system, simulating device errors, exhausting essential resources, breaking VMScluster communication channels, and creating excessive I/O or locking loads.

The fault loads that Faulty Towers provided were predictable and quite repeatable. When problems occurred during test execution, the precise fault loads involved could be readily reproduced to accelerate the process of diagnosing the underlying defect and verifying a proposed fix. Faulty Towers also provided a means of easily tailoring, controlling, and monitoring the automatic insertion of faults during our profile testing. The result was better coverage of error paths using a much simpler test environment.

Staging Test Implementation To stage the introduction of profile complexity, we gradually increased the number of error events applied to successive groupings of test profiles. We began our testing with a simple base profile to which further load complexity could be progressively added. This base profile involved only three two-member shadow sets with just one of the targeted error events occurring during each run. System load was limited to reads and writes across the shadow sets. No system disks were shadowed in this base profile.

During the initial execution of the base profile, we tested resource exhaustion. With each subsequent round of testing, we systematically incorporated additional complexity: more test configurations, three-member shadow sets, shadowed system disks, complex error profiles, and system-wide loading.

Operational Sequence for Profile Test Execution Another important aspect of the profile testing was the use of a prescribed operational sequence during profile test execution. This sequence is shown in Figure 9.

Profile test runs began with the mounting of a single shadow set member. The addition of a second or third member caused the initiation of a copy operation from the existing member to the added device(s). The removal of a VMScluster node that had a shadow set mounted would cause shadowing to initiate a merge operation on the shadow set. To maintain consistency across our test runs, we would manually add back into a shadow set any member(s) that were expelled due to the node removal. At this point, shadowing is expected to progress sequentially through copy and merge operations to create a fully consistent shadow set.

The I/O required by these copy and merge operations formed the base load on the system. User I/O to the shadow sets incremented the effective load on the system as did the disruption of I/O due to error events. During the period when user I/O and error events were sustained at their heaviest levels, I/O for copy operations could stall entirely. Winding down error insertion and user I/O enabled copy

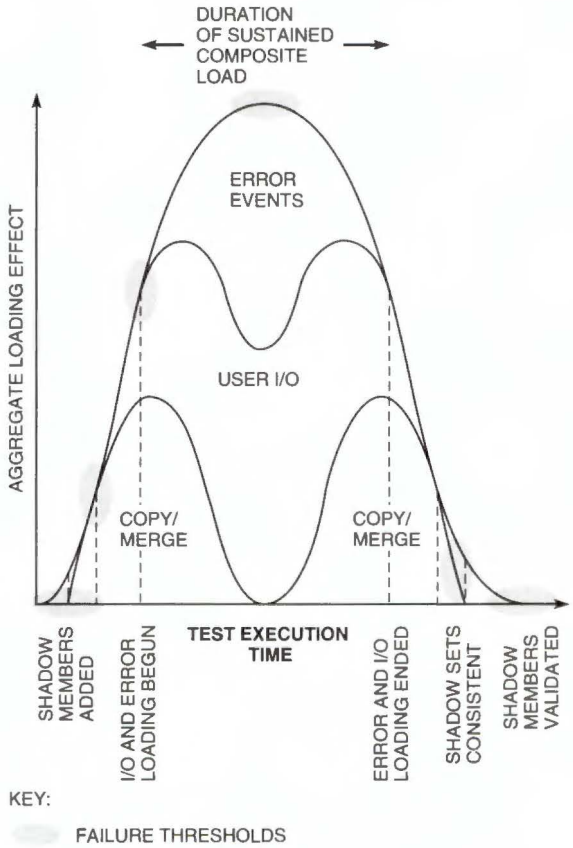


Figure 9 Operational Sequence for Profile Testing

and merge operations to complete. At that time, the shadow sets could be dismounted and the individual members compared for data consistency.

During the test execution sequence, each step represented a new threshold at which failures were more likely to occur. As testing continued and the shadowing code stabilized, early test execution sequences tended to generate fewer new defects. To find additional defects, we increased the complexity of the test execution sequence, fault loads, and configurations. The result was a sustained effectiveness in defect detection throughout our profile testing.

Project Results

The process described above enabled us to satisfy both the quality and schedule goals of our project to port shadowing to the OpenVMS AXP system. How significantly the process contributed to this accomplishment is shown below with data describing our improvements in process, product quality, and productivity.

Improving Process

Inspections and profile testing were our two key process enhancements. Data that tracked defect detection and product stabilization during these steps underscores their contribution.

Tracking Defect Detection The solid line in Figure 10 shows defect detection by week throughout the life of the project. Note that the solid line in the figure tracks very closely and, at times, overlaps the dashed line used to indicate the inspection time. Only high severity defects that resulted in a code change are represented by these defect counts. The time period before January 4 involved neither inspections nor testing; the time period after June 14 involved only testing. During March, porting work stopped due to team members being temporarily reassigned to critical development tasks in support of OpenVMS AXP version 1.5. Allowing for that gap in March, the trend in defect detection from both inspections and testing exhibited a steady decline from mid-January through October. This trend provides a strong indication that the project was on schedule and not deferring the bulk of defect removal to the latter, more costly stages of development.

The dashed line in Figure 10 shows the amount of time spent weekly in inspections. It shows that the highest rates of defect detection resulted from, and were in rough proportion to, time spent in inspections. Early defect removal using inspections represented a marked change from traditional practices within OpenVMS Engineering.

Tracking Product Stabilization The manner in which we designed and implemented profile testing gave rise to a pair of metrics for tracking both test effectiveness and product stabilization by tracking test execution results. The first of these metrics was a ratio between the test execution time as measured in days of test execution per VMScluster node (node-days) and the number of problem reports that resulted. The second was a ratio between the number of problem reports submitted to development engineers and the number of defects that were detected as a result.

Figure 11 shows these two metrics plotted weekly during the course of our profile testing. The key to interpreting these trends lies in contrasting the two ratios. For example, low node-days/problem report accompanied by high problem-reports/defect in July 1993 indicates test execution errors as we learned how to load and test shadowing in a

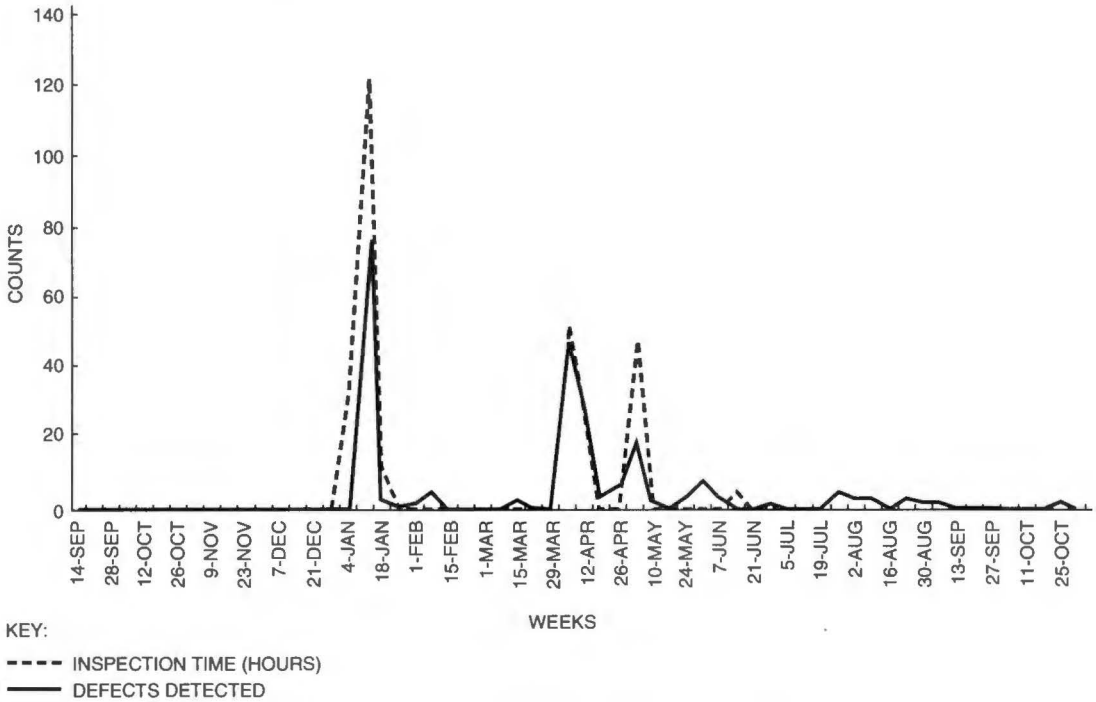


Figure 10 Inspection Time and Defect Detection

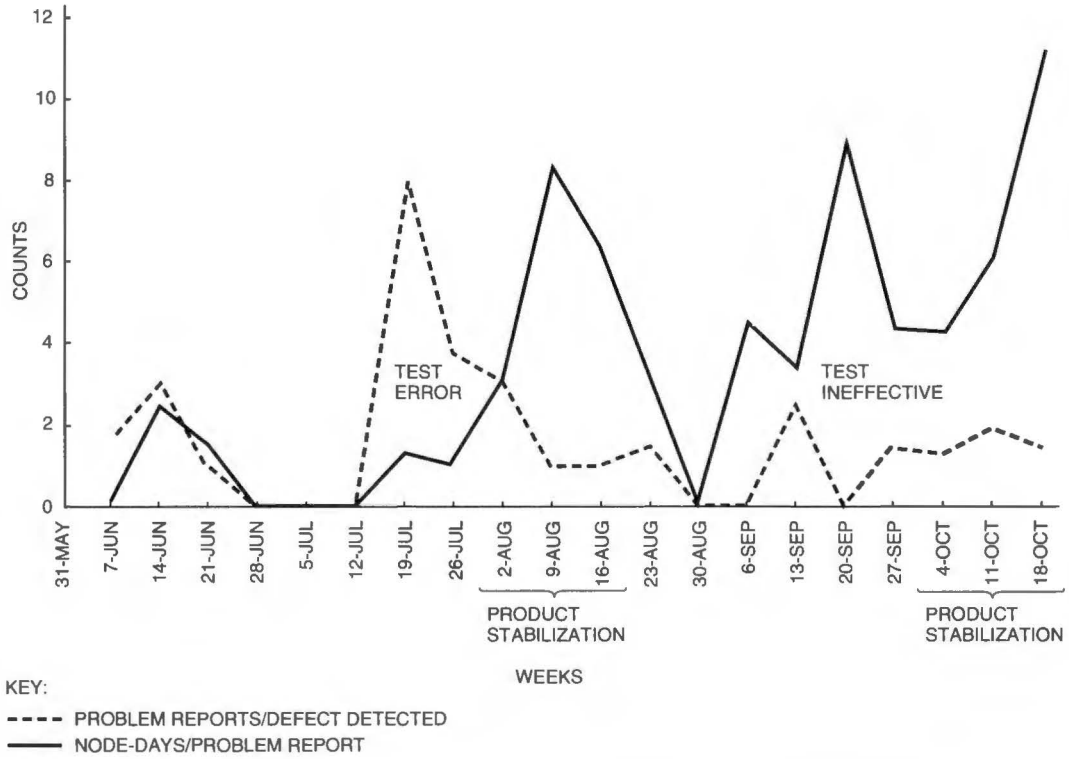


Figure 11 Metrics for Evaluating Test Effectiveness and Product Stabilization

VMScluster system. In late September, high node-days/problem report with no defects indicates the execution of an ineffective test profile. Near-zero curves in early July and early September indicate when vacations were interrupting our testing.

During the two product stabilization periods indicated in Figure 11, the increase in node-days/problem report accompanied by a near one-to-one ratio between problem reports and defects indicates that the product was stabilizing in spite of sustained test effectiveness. The first period preceded beta test; the second preceded product release.

Assuring Quality

Containing and removing defects were at the core of our release criteria. As Figure 12 shows, actual defect removal during the project exceeded projections by 8 percent. By selecting our defect goal based on industry-standard defect densities and exceeding this goal, we have assured quality in our product. Of the total defects actually removed, 232 were in the shadowing code. This suggests a defect removal rate of 12 defects per 1,000 NCSS for the shadowing code base, which is consistent with industry data reported by Schulmeyer.[19]

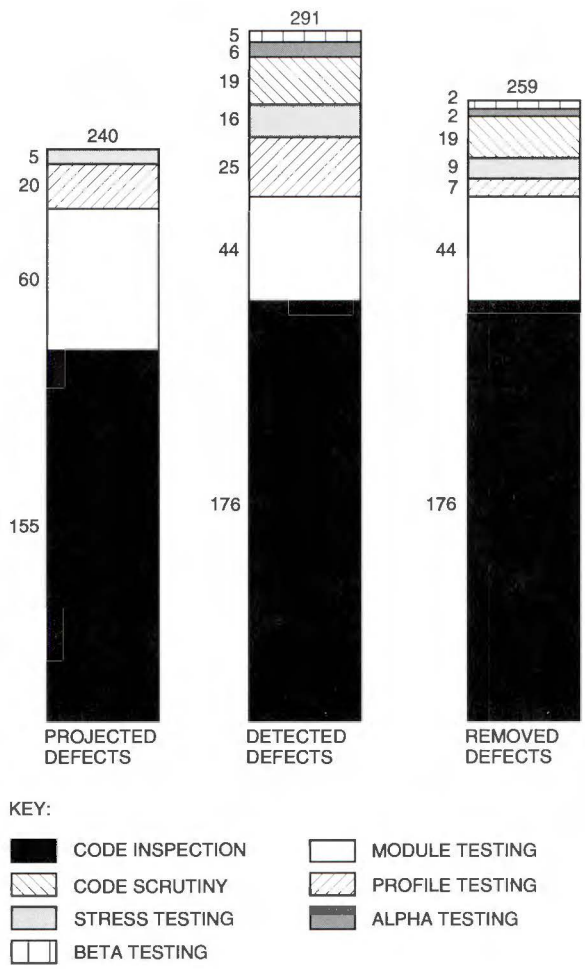


Figure 12 Comparison of Projected and Actual Defect Detection and Removal by Method

Of the 176 defects removed through inspections, 43 were found in the nonported shadowing code. This suggests a defect removal rate for the ported code of 53 per 1,000 NCSS, which again falls within the range reported by Schulmeyer. The combination of testing and inspections resulted in the removal of 59 defects from the unmodified shadowing code base. This represents a net reduction of 3.5 defects per 1,000 NCSS within the unmodified shadowing code as a result of the port!

The significance of this level of defect removal as an indicator of high quality was corroborated in two ways. First, all release criteria were satisfied prior to releasing the ported shadowing product. Second, the character of defects detected during the testing phase of the project changed. Whereas most defects detected in early test runs were introduced during the port, virtually all defects detected in later test runs were residual in the underlying code base. Again, removing this latter type of defect meant better overall quality for customers running mixed-architecture VMScluster systems.

Another aspect of quality was identifying those defects that should not be fixed but only contained. Because this project had very limited scope, duration, and resources, we had to carefully evaluate changes that could destabilize the entire product and jeopardize its overall quality. The comparison of detected to removed defects in Figure 12 shows that several problems fell into this category. Many of these defects were triggered by system operations that exceeded the fundamental design limitations of the product. Some occurred in obscure error paths that could only be exercised with our new test methods. Others were due to new combinations of hardware possible in mixed-architecture VMScluster systems. In each of these instances, we assured that the scope of the defect was limited, its frequency low, and its impact predictable. When necessary, we constrained supported hardware configurations or system behavior so that the defect could not cause unrecoverable failures. Finally, we fed back our analyses of these

defects into the ongoing shadowing support effort so that they could be removed in future releases through appropriate redesign.

Increasing Productivity

The following data shows how our enhanced process contributed to the quality. This data shows the relative cost-effectiveness, and hence improved productivity, achieved through inspections and profile testing.

Engineering Costs The proportion of total engineering hours expended during each step of the project is depicted in Figure 13. This figure indicates that only 15 percent of the hours (inspections and module testing) resulted in the removal of 85 percent of all defects. Inspections alone accounted for only 5 percent of the engineering hours but 68 percent of the defect removal!

The actual cost for removing defects by inspection averaged 1.7 engineer hours per defect. During module testing, when engineers worked individually to debug the functions they had ported, the cost of defect removal jumped to 15 engineer hours per defect. During integration testing, where the entire shadowing driver was tested in a complex environment, an average of 85 engineer hours was spent per defect exclusive of the time spent to execute the tests. Clearly, removing the bulk of the defects from the ported code prior to beginning testing of the integrated shadowing product dramatically reduced both the cost and time required for defect removal.

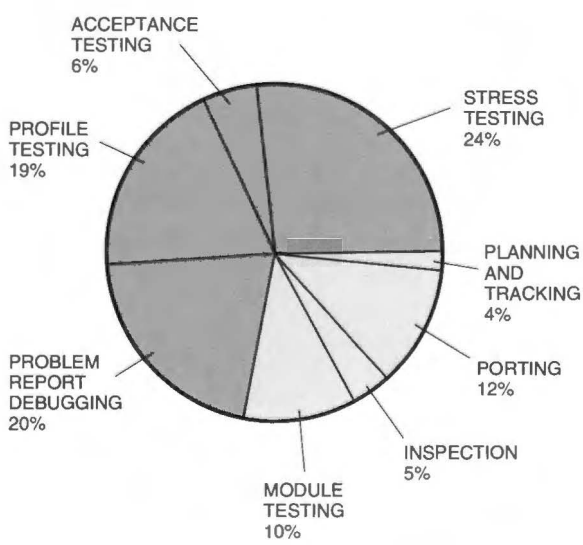


Figure 13 Distribution of Total Engineering Hours by Process Step

Defect Yield Assuming a 95 percent overall defect yield for shadowing prior to release, the relative yield of inspections during this project was 65 percent. When calculated against defects found only in shadowing, the yield for inspections jumps to 75 percent—a very high removal efficiency when compared with industry data.[8] Relative defect yield was consistent with industry data for module testing at 16 percent, but low for profile and stress testing at a combined value of 6 percent. Given the high engineering cost shown above for removing defects during integration test of shadowing, this is in fact quite a favorable result.

Test Cost-effectiveness As Figure 13 indicates, testing remained the most costly portion of the project. Executing and debugging problems from roughly 86,000 node-hours of stress, performance, and profile testing accounted for 69 percent of the project's total engineering hours. In fact, the ratio of engineer hours expended for test versus implementation was roughly 1.8 times higher than Grady reports for 48 projects involving systems software.[20] Given the complexity of the VMScluster systems historically required to test shadowing, this ratio is no surprise.

Nevertheless, our project's results indicate that profile testing was significantly more cost-effective than large-scale stress testing for detecting defects. Profile testing's overall ratio of test engineer days per defect was 25 percent better at 6.2 days than stress testing's 8.3 days. Moreover, profile testing's overall ratio of machine test time per defect was more than an order of magnitude better at 7.4 node-days than stress testing's 95.2 node-days!

This improvement in cost-effectiveness was achieved with no loss in defect removal capability. When compared with large-scale stress testing, profile testing of shadowing proved equally effective overall at detecting defects. During the beta test period for shadowing, each of these test methods accounted for roughly 20 percent of the defects detected when allowing for duplication between methods. The number of problem reports per defect was also comparable with ratios of 2.4 for profile testing and 2.0 for stress testing.

Figure 14 contrasts the cost-effectiveness of each method of defect detection employed during the validation of shadowing. Note that because this chart uses a log scale, any noticeable difference between bar heights is quite significant. This chart bears out

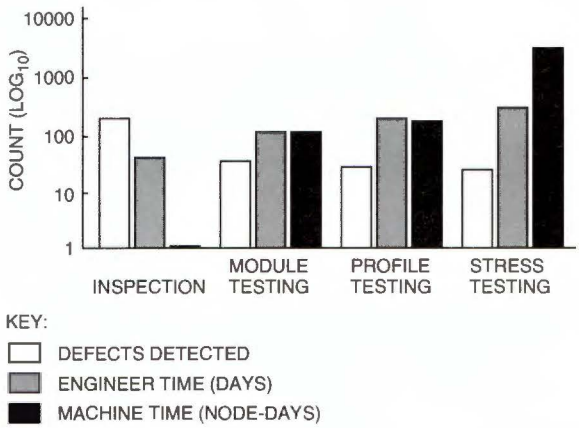


Figure 14 Relative Cost-effectiveness of Defect Detection Methods

conventional wisdom on defect removal: detection prior to integration through the use of inspections and module testing is by far the most cost-effective. It also suggests that profile testing has a cost-effectiveness on the same order of magnitude as module testing, while providing the same defect detection effectiveness as large-scale stress testing.

Conclusions

At the outset of the shadowing port, given its unique challenges, we believed that the existing development process for OpenVMS would not enable us to meet the project's goals. By taking charge of our engineering process, however, we not only met those goals but also demonstrated that changes to the established process could result in higher productivity from our engineering resources and better quality in the delivered product.

The volume shadowing port from OpenVMS VAX to OpenVMS AXP was successful in meeting an aggressive schedule and in delivering a stable, high-quality product. There were two key process innovations that led to our success. The first was the use of inspections to detect and remove a large percentage of the porting defects before any investment in testing. By finding a majority of the defects (68 percent) at the lowest possible cost (1.5 hours per defect), fewer overall resources were required.

The second key innovation was the use of profile testing to cover a very large and complex test domain. Having a test strategy that used well-defined and repeatable tests to target problem areas in shadowing code allowed us to efficiently find defects and verify fixes. With profile testing, we managed to achieve the defect detection effectiveness of large-scale integration testing at the same relative cost as module testing.

The results of our process innovations would not have been realized if we had waited for our organization's process to evolve up the CMM levels. By changing the engineering process of a small team, we delivered high-quality software on schedule and at a lower cost.

Future Developments

As a result of our project's accomplishments, OpenVMS Engineering is giving serious consideration to our practices. Many groups are acknowledging the gains possible when formal inspections are used to contain defects. Moreover, the organization's problem reporting system is being upgraded to include mechanisms for tracking defects that incorporate many of the fields used in our defect tracking database.

OpenVMS Engineering is also evaluating further use of the profile testing methodology in its testing efforts. As Figure 13 indicated, improving the effectiveness of our integration testing may offer the most significant opportunity for reducing engineering costs and accelerating the schedules of our software releases. Since experimental design principles were used in the creation of the tests, statistical evaluation of the results is a potentially exciting opportunity. An analysis of variance that explores the relationship between test factors and defects could indicate which test factors contribute significantly to defect detection and which do not. This could give us a clear statistical indication of where to direct our testing efforts and development resources.

Acknowledgments

The success and the final form of the our software development process were a direct consequence of the diverse skills and unflagging efforts of the shadowing team members involved in its implementation: Susan Azibert, Kathryn Begley, Nick Carr, Susan Carr, Mary Ellen Connell, Tom Frederick, Paula Fitts, Kimilee Gile, Greg Jordan, Bruce Kelsey, Maria Leng, Richard Marshall, Kathy Morse, Brian Porter, Mike Stams, Barbara Upham, and Paul Weiss. Support from Howard Hayakawa, manager of the VMScluster group, provided the opportunity for us to initiate such extensive process enhancements in the context of a critical project for OpenVMS.

References

1. S. Davis, "Design of VMS Volume Shadowing Phase II—Host-based Shadowing," *Digital Technical Journal,* vol. 3, no. 3 (Summer 1991): 7–15.

2. *Volume Shadowing for OpenVMS* (Maynard, MA: Digital Equipment Corporation, November 1993).

3. M. Paulk, B. Curtis, M. Chrissis, and C. Weber, *Capability Maturity Model for Software V1.1* (Pittsburgh, PA: Carnegie-Mellon University, Software Engineering Institute, Technical Report, CMU/SEI-93-TR-24 ESC-TR-93-177, February 1993).

4. W. Humphrey, *Managing the Software Process* (Reading, MA: Addison-Wesley Publishing Company, 1989/1990).

5. R. Grady and D. Caswell, *Software Metrics: Establishing a Company-Wide Program* (Englewood Cliffs, NJ: Prentice-Hall, 1987): 112.

6. *VMScluster Systems for OpenVMS* (Maynard, MA: Digital Equipment Corporation, March 1994).

7. W. Perry, *Quality Assurance for Information Systems: Methods, Tools, and Techniques* (Boston: QED Technical Publishing Group, 1991): 541–563.

8. C. Jones, *Applied Software Measurement* (New York: McGraw-Hill, Inc., 1991): 174–176, 275–280.

9. N. Kronenberg, T. Benson, W. Cardoza, R. Jagannathan, and B. Thomas, "Porting OpenVMS from VAX to Alpha AXP," *Digital Technical Journal,* vol. 4, no. 4 (Special Issue 1992): 111–120.

10. R. Pressman, *Software Engineering: A Practitioner's Approach* (New York: McGraw Hill, 1992): 334–338.

11. M. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal,* vol. 15, no. 3 (1976): 182–211.

12. D. Freedman and G. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products* (New York: Dorset House Publishing, 1990).

13. J. Musa, "Operation Profiles in Software Reliability Engineering," *IEEE Software* (New York: IEEE, March 1993): 14–32.

14. T. McCabe and C. Butler, "Design Complexity Measurement and Testing," *Communications of the ACM,* vol. 32, no. 12 (December 1989): 1415–1425.

15. G. Taguchi and M. Phadke, "Quality Engineering Through Design Optimization," *Conference Record,* vol. 3 (IEEE Communications Society GLOBECOM Meeting, November 1984): 1106–1113.

16. T. Pao, M. Phadke, and C. Sherrerd, "Computer Response Time Optimization Using Orthogonal Array Experiments," *Conference Record,* vol. 2 (IEEE International Communications Conference, June 1985): 890–895.

17. S. Schmidt and R. Launsby, *Understanding Industrial Designed Experiments* (Colorado Springs: Air Academy Press, 1989): 3-1–3-32.

18. *SAS/QC Software, Version 6* (Cary, NC: SAS Institute, Inc., 1992).

19. G. Schulmeyer, *Zero Defect Software* (New York: McGraw-Hill, Inc., 1990): 73.

20. R. Grady, *Practical Software Metrics for Project Management and Process Improvement* (Englewood Cliffs, NJ: Prentice-Hall, 1992): 42.

David G. Conroy
Thomas E. Kopec
Joseph R. Falcone

The Evolution of the Alpha AXP PC

The DECpc AXP 150 personal computer is not only the first in Digital's line of Alpha AXP PC products but also the latest in a line of experimental low-cost systems. This paper traces the evolution of these systems, which began several years ago in Digital's research and advanced development laboratories. The authors reveal some of the reasoning behind the engineering design decisions, point out ideas that worked well, and acknowledge ideas that did not work well and were discarded. Chief among the many lessons learned is that combining Alpha AXP microprocessors and industry-standard system components is within the abilities of any competent digital design engineer.

The DECpc AXP 150 system is Digital's first Alpha AXP personal computer (PC) product that supports the Microsoft Windows NT operating system. This product is the latest member of an evolutionary series of low-cost systems that take advantage of PC components and standards. Work on these systems began several years ago in Digital's research and advanced development laboratories.[1] By tracing the evolution of the Alpha AXP PC from the Beta demonstration system (which pioneered the concept of the Alpha AXP PC) through the Theta system (which incorporated an Extended Industry Standard Architecture [EISA] bus) to the DECpc AXP 150 product, this paper shows how experimental systems solved many problems in anticipation of products.

The Alpha AXP PC design philosophy is summed up by one of the DECpc AXP 150 advertising slogans: It's just a PC, only faster. By being culturally compatible with industry-standard PC systems, Alpha AXP PC systems can exploit the huge infrastructure of low-cost component suppliers supported by the high volumes of the PC marketplace and be cost competitive in that marketplace.

Alpha AXP PC systems typically include little functionality in the base system. Many additional capabilities are provided by option cards via the Industry Standard Architecture (ISA) or EISA bus. Such capabilities can be upgraded easily to keep pace with technological developments. In addition, the increasing fragmentation of the desktop computer market has made it virtually impossible to design a single product that addresses the needs of all market segments. By providing option slots, systems can be configured to meet a wide variety of customer requirements.

Beta Demonstration System

In the early days of the Alpha AXP program (1990), conventional wisdom said that the DECchip 21064 microprocessor could not be used to build low-cost computer systems—it consumed too much power, was difficult to cool, and was optimized for large, high-performance systems.[2,3] Therefore, Digital was not designing any low-cost systems; all product groups were waiting for a low-cost Alpha AXP CPU, the DECchip 21066 microprocessor (which was announced in September 1993).

In December 1990, several members of the Semiconductor Engineering Advanced Development Group (the same group that began the Alpha AXP architecture development and designed the DECchip 21064 microprocessor) decided to investigate the feasibility of producing low-cost systems. In February 1991, with the help of Digital's Cambridge Research Laboratory, they began designing and building a demonstration system called Beta.

Although the Beta system used the same enclosures, power supplies, expansion bus option cards, and peripherals as industry-standard PCs, true PC compatibility was never a goal. The intent was simply to demonstrate the feasibility of building low-cost systems; standard PC components were used because doing so eliminated some design work.

Hardware Design

Figure 1 is a block diagram of the Beta demonstration system. The hardware design was completely synchronous. The DECchip 21064 CPU operated at 100 megahertz (MHz) and generated the 25-MHz clock that ran most of the logic. The ISA bus

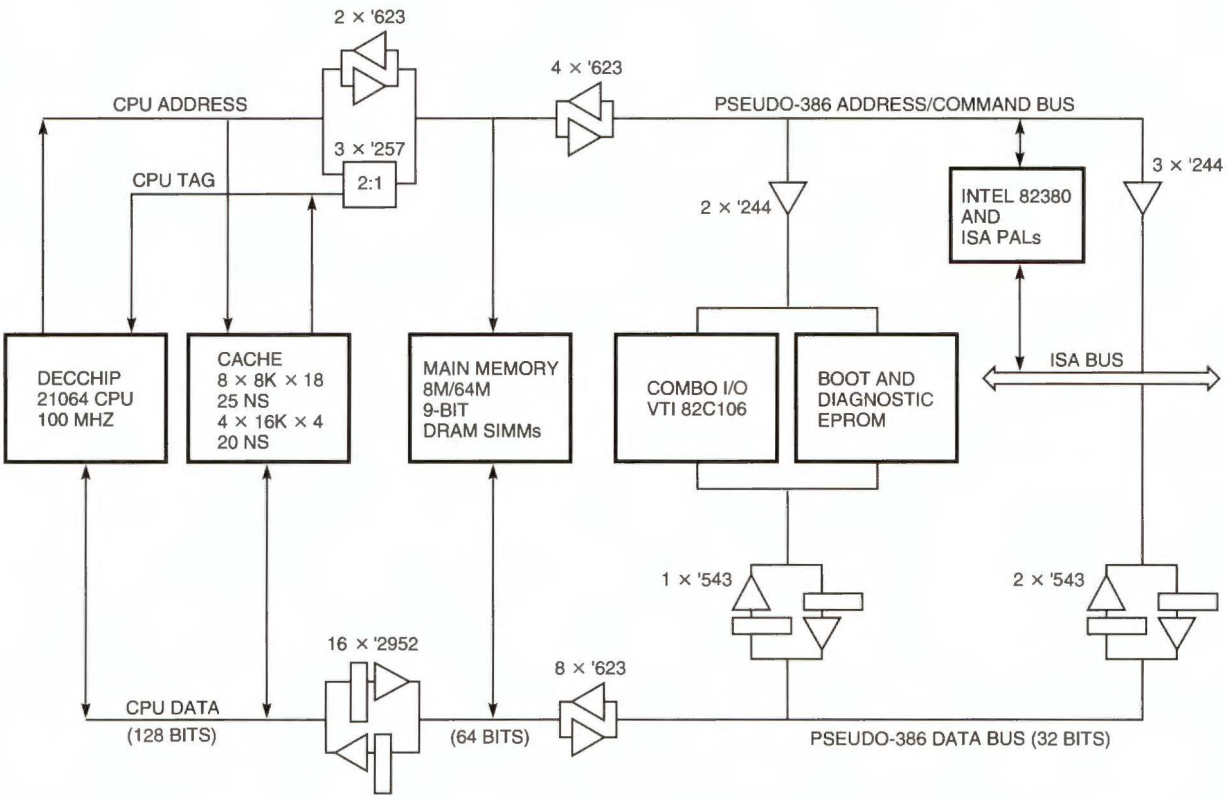


Figure 1 Block Diagram of the Beta Demonstration System

interface operated at 8.33 MHz, generated by dividing the 25-MHz clock by three.

The Beta system was packaged in an attractive, low-profile, tabletop enclosure. The compact size forced the physical arrangement to be cramped, which made cooling the CPU more difficult. Because the CPU operated at 100 MHz and thus dissipated only about 16 watts, it could be cooled by a large, aluminum heat sink (6.9 centimeters [cm] wide by 8.1 cm long by 2.5 cm deep) and the 20 linear centimeters per second of available airflow.

A three-terminal linear regulator produced the required 3.3-volt power. Cooling this regulator was almost as difficult as cooling the CPU chip but was accomplished with an off-the-shelf heat sink and the available airflow.

The backup cache and memory systems in previous Alpha AXP designs were absolute-performance driven, that is, designed to meet specific performance requirements. Usually, these requirements were set at the level of performance needed to outperform competitors. Workstation performance requirements tended to emphasize the SPEC benchmark suite.

The Beta system designers took a different approach. They proposed various backup cache and memory systems, estimated the performance and cost of each one, and selected the one that had the best performance per unit cost (beyond a minimum absolute-performance requirement).

Ultimately, the Beta demonstration system used a 128K-byte (128KB) write-back cache that was 128 bits wide and built with the 25-nanosecond (ns), 8K-by-18-bit static random-access memories (SRAMs) designed to be used with the Intel 82385 cache controller. At 25 ns, the SRAMs were slow, but by using 48-milliampere (mA) address drivers, incident wave-switching, and AC parallel termination, it was possible to both read and write the cache in 40 ns (four CPU cycles).[4] Writing the cache SRAMs was simplified by the fact that, in addition to the normal system clock (sysclk1), the CPU chip generated a delayed system clock (sysclk2). This delayed clock had exactly the right timing to be used as a cache SRAM write pulse. Enabling logic, built using a 7.5-ns programmable array logic (PAL) device, ensured that the delayed clock was sent to the cache SRAM only when a write pulse was actually

needed. Variations on this design, shown in more detail in Figure 2, have been used in several other Alpha AXP systems, including the EB64 (an evaluation board for the DECchip 21064 microprocessor) and an experimental DECstation 5000 Model 100 daughter card, as well as the Theta and DECpc AXP 150 systems.

Main memory used 8 or 16 standard, 9-bit-wide, dynamic random-access memory (DRAM), single in-line memory modules (SIMMs). Although the cache was 128 bits wide, the memory was only 64 bits wide and cycled four times, in page mode, to deliver a 32-byte cache line. Main memory was protected by 32-bit longword parity that was generated and checked by the CPU and copied without interpretation to and from the backup cache. The memory controller was built from PALs clocked at 25 MHz. Registered devices generated all the control signals, so none of the PALs needed to be very fast. The long minimum delay of these slow PALs made clock skew management easy.

The fact that the I/O system needed to be able to perform partial longword direct memory access (DMA) writes complicated the backup cache and memory system because both the cache and memory were protected by longword parity. After rejecting the options of (1) storing byte parity in memory with no parity in the cache and (2) performing read-modify-write cycles on partial longword writes, the designers selected the Intel 82380 multifunction peripheral chip. This chip can perform longword assembly and disassembly for narrow DMA peripherals. Partial longword DMA writes never worked properly, however, because this chip did not function exactly as described in its data sheet. The chip was not used in future designs.

The I/O system was built around an I/O bus that imitated the signaling of an Intel386 DX CPU chip (since the Intel 82380 device was compatible with the Intel386 DX chip). Logic between the memory bus and the I/O bus translated DECchip 21064 cycles into cycles that mimicked the style of Intel386 DX cycles. Figure 3 illustrates the translation.

Complete imitation of the signaling of an Intel CPU requires the generation of memory-space reads and writes with byte granularity, I/O-space reads and writes with byte granularity, and several special-purpose cycles. However, only a small subset of these cycles (exactly eight) actually needed to be generated in the Beta system. Thus, DECchip 21064 address bits [32..30] were used to encode the details of the cycle; a single PAL expanded these 3 bits into the Intel cycle-type signals. An alternative scheme that stored unencoded Intel cycle-type signals in a control register was rejected because the control register would have to be saved and restored in interrupt routines.

The external interface of the DECchip 21064 microprocessor is strongly biased toward the reading and writing of 32-byte cache lines; external logic has difficulty gaining access to DECchip 21064 address bits [04..03] and cannot access address bit [02]. Therefore, the Beta system used DECchip 21064 address bits [29..27] to supply Intel address

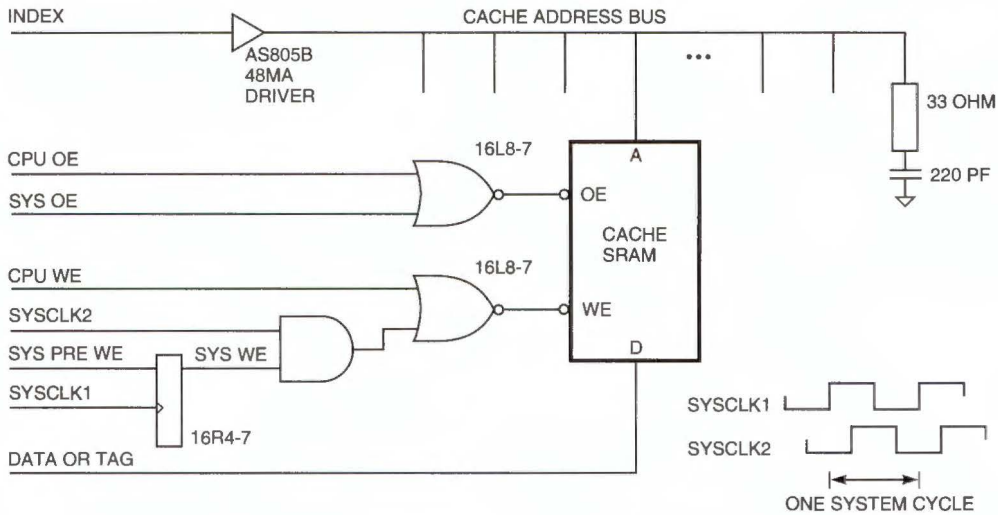


Figure 2 Details of the Beta System Cache

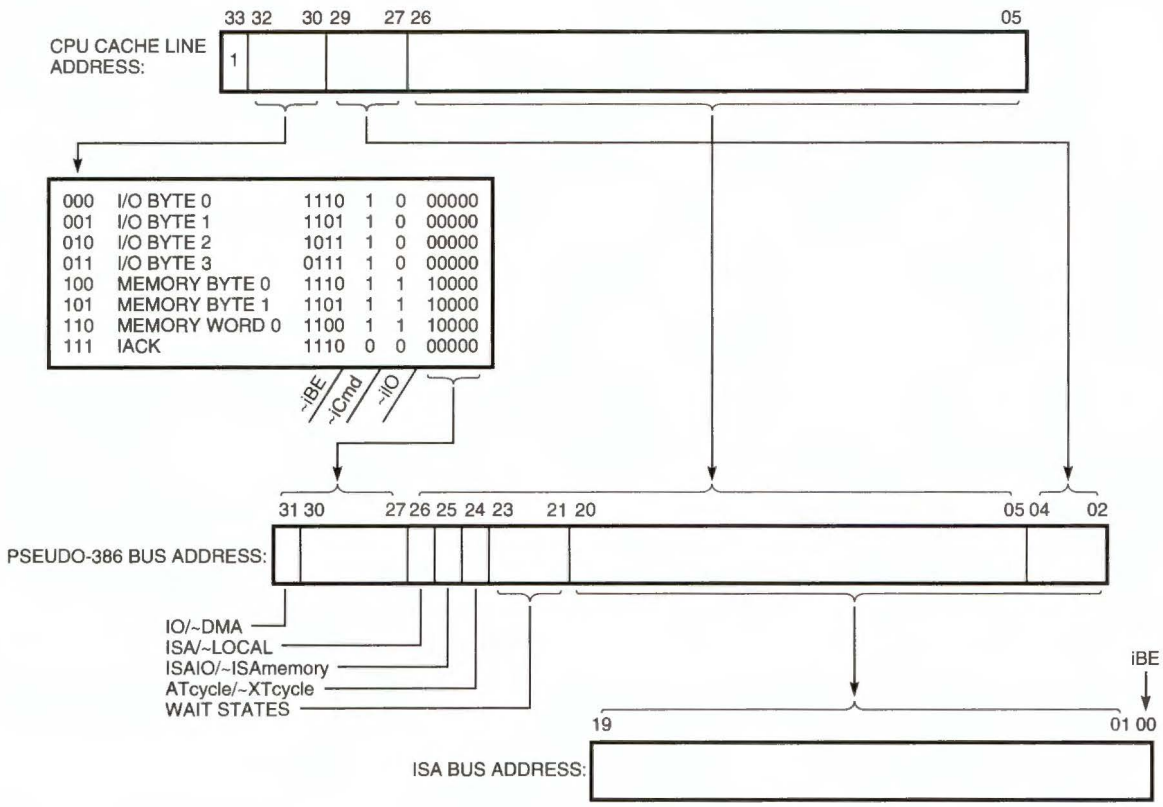


Figure 3 Beta System I/O Addressing

bits [04..02]. This odd positioning, which was chosen because it saved a few parts in the address path, seemed harmless; the CPU was so much faster than the I/O that repositioning the low-order address bits did not reduce Beta system performance. It was a bad trade-off, however, because the odd addressing scheme made writing low-level software for devices containing buffers (e.g., a PC-style video adapter) painful and error prone.

The ISA bus interface connected to the I/O bus. Since all DMA control was inside the Intel 82380 chip, the ISA interface functioned as a simple slave that translated Intel386 DX cycles into ISA cycles.[5] Once again, the Beta system used address bits to encode the details of the cycles. Many programmable options were included because the sensitivity of ISA cards to bus timing was unknown. In fact, the ISA cards that were used could tolerate wide variations in bus timing and never required the unusual bus cycle options.

The DMA controllers in the Intel 82380 handled ISA-bus DMA transfers and generated the appropriate read and write requests. The cache and memory system responded with the appropriate read and write cycles. The refresh controller in the Intel 82380 handled memory refresh and generated the appropriate refresh requests. The memory system responded with column address strobe-before-row address strobe (CAS-before-RAS) refresh cycles. In both cases, the Beta system used the CPU's holdReq/holdAck protocol (in which the CPU stops and tristates most of its pins) to avoid conflicts between the CPU and DMA or refresh on the cache and/or memory system.

During DMA read cycles, the backup cache read data (in anticipation) and performed the tag compare operation in parallel with the RAS-to-CAS delay of the DRAMs. If the reference was a miss, then CAS was asserted and the data came from the DRAMs. If the reference was a hit, then the buffers between the cache data bus and the memory data bus were enabled and the data came from the cache; the cycle on the DRAMs became a RAS-only refresh cycle.

During DMA write cycles, the data was written to the DRAMs. The cache performed the tag compare operation in parallel with the RAS-to-CAS delay of

the DRAMs. If the reference was a hit, the data was written into the backup cache as well (without changing the state of the dirty bit), and the appropriate internal CPU cache line was invalidated.

The local peripheral interface also connected to the I/O bus. This interface included (1) an 82C106 PC Combo I/O chip from VLSI Technology, Inc., with a keyboard interface, a mouse interface, two serial lines, a parallel printer port, a time-of-year clock, and a small SRAM with battery backup; (2) a control register that contained a few random control bits; and (3) a 64KB erasable programmable read-only memory (EPROM) that contained the boot and console code. Code in the DECchip 21064 serial read-only memory (ROM) copied the EPROM to main memory for execution, thus eliminating the need for hardware that could read 32 bytes from the byte-wide EPROM and assemble the data into a cache line.

The hardware was completed in a very short time—about eight weeks from the start of the design to releasing the databases to the printed circuit board and assembly houses. One person did the logic design, the logic simulation, and the timing verification. A second person designed the physical layout and solved all the mechanical and cooling problems. A set of quick-turnaround, computer-aided design (CAD) tools developed by Digital's Western Research Laboratory in Palo Alto, California, allowed concurrent design verification (with modifications) and physical design.

Software Design

The Beta system was debugged using a simple console program, which drove an ordinary terminal plugged into one of the serial ports and allowed the designers to peek and poke at memory and I/O devices. Both the hardware and the simple console program were debugged in one day, and most of that day was spent looking for a single software bug in the CPU chip initialization code.

The designers augmented the simple console program to perform more extensive diagnostics, drive a standard Intel PC keyboard and ISA bus display, load programs by name from a file system on a small computer systems interface (SCSI) disk using a standard ISA bus disk controller, and load programs by name via BOOTP over the Ethernet using a standard ISA bus network controller. All this code fit into the 64KB EPROM, albeit compressed.

Eventually, the designers built a fairly complete version of the UNIX operating system for the Beta system, starting from the port of the Berkeley Software Development (BSD)-based ULTRIX system built for the original Alpha AXP Development Unit.[6] This UNIX system included a port of X11R5, believed to be the first 64-bit X11 server ever built.[7]

Performance

The CPU performance of the Beta system was estimated to be 70 for integer (SPEC dhrystone, eqntott, espresso) and 65 for floating point (SPEC fpppp, matrix300, tomcatv). A system with a 128-bit-wide cache and a 128-bit-wide memory would have had a performance of 70 integer and 75 floating point. A system with a 64-bit-wide cache and a 64-bit-wide memory would have had a performance of 65 integer and 55 floating point. All these estimates were obtained from a trace-driven performance model of the DECchip 21064 CPU, the backup cache, and the memory system.

The ISA bus limited the I/O performance of the Beta system to approximately 4 megabytes per second (MB/s). Thus, Beta is the most unbalanced Alpha AXP system ever designed.

Outcome

Digital ultimately built 35 Beta demonstration systems, which were used in many presentations, including a stockholders' meeting and a private demonstration for Bill Gates at Microsoft Corporation. The Beta machines were proof that the DECchip 21064 microprocessor could be used to build low-end systems and prompted a number of low-end system projects to be started. Beta systems were used in the early development of the Alpha AXP version of the Windows NT operating system.

Although many aspects of the Beta design worked well, dealing with the PC option cards did not go smoothly. The designers knew that there was not much low-level programming documentation available and took care to select option cards based on very large-scale integration (VLSI) chips for which good documentation could be obtained. They were often astounded, however, at how difficult these devices were to program and how often they did not work exactly as described. By the end of the project, it was clear to the designers that they could use PC components to build interesting systems. It was equally evident that PC option cards could be difficult to use, since little or none of the low-level software provided by the option cards' supplier (basic I/O system [BIOS] ROM code, MS-DOS device drivers, or user code that manipulated the option card directly) was usable. Video graphics array

(VGA) cards proved to be particularly troublesome, since they tended to conform to the VGA programming standard only after code in their BIOS ROM performed a module-specific initialization sequence.

Theta System

The next step in the evolution of the Alpha AXP PC took place in the winter and spring of 1992 with the design of the Theta system. This machine was based on the Beta design but used the EISA bus, driven by the Intel 82350DT chip set.[8] Figure 4 is the block diagram of the Theta system. In addition to replacing the ISA I/O system with an EISA I/O system, the Theta design stored its firmware in flash EPROM and used an I/O bus that imitated the signaling of an Intel486 DX CPU chip.

The Theta designers made several mistakes because they lacked a complete understanding of the nuances of industry-standard PC architecture. For example, the EISA bus uses special memory read and write signals when accessing the first 1M byte (1MB) of memory. The designers were not aware that one chip in the EISA chip set was checking address ranges for another chip in the set. Consequently, they did not realize that the address map they had chosen for their system would cause the range check to fail. This failure would block the generation of the special memory read and write signals and thus make the buffer memory in a network card inaccessible. Further, it is not clear that the Theta designers could have determined that this would occur from the chip set data sheet alone.

Digital built few Theta systems, and little system software ever ran on these systems. The Theta project, however, provided an environment for learning how to use the Intel 82350DT EISA chip set in an Alpha AXP system. The project team developed an extensive set of exercisers for popular EISA cards, which were very useful in the development of future systems.

DECpc AXP 150 Product

The charter for the design and manufacture of the DECpc AXP 150 product, code-named Jensen, belonged to the Entry Level Solutions Business in Ayr, Scotland, which had previously designed the successful line of MicroVAX 3100 systems. However, Digital felt that it was important to incorporate the

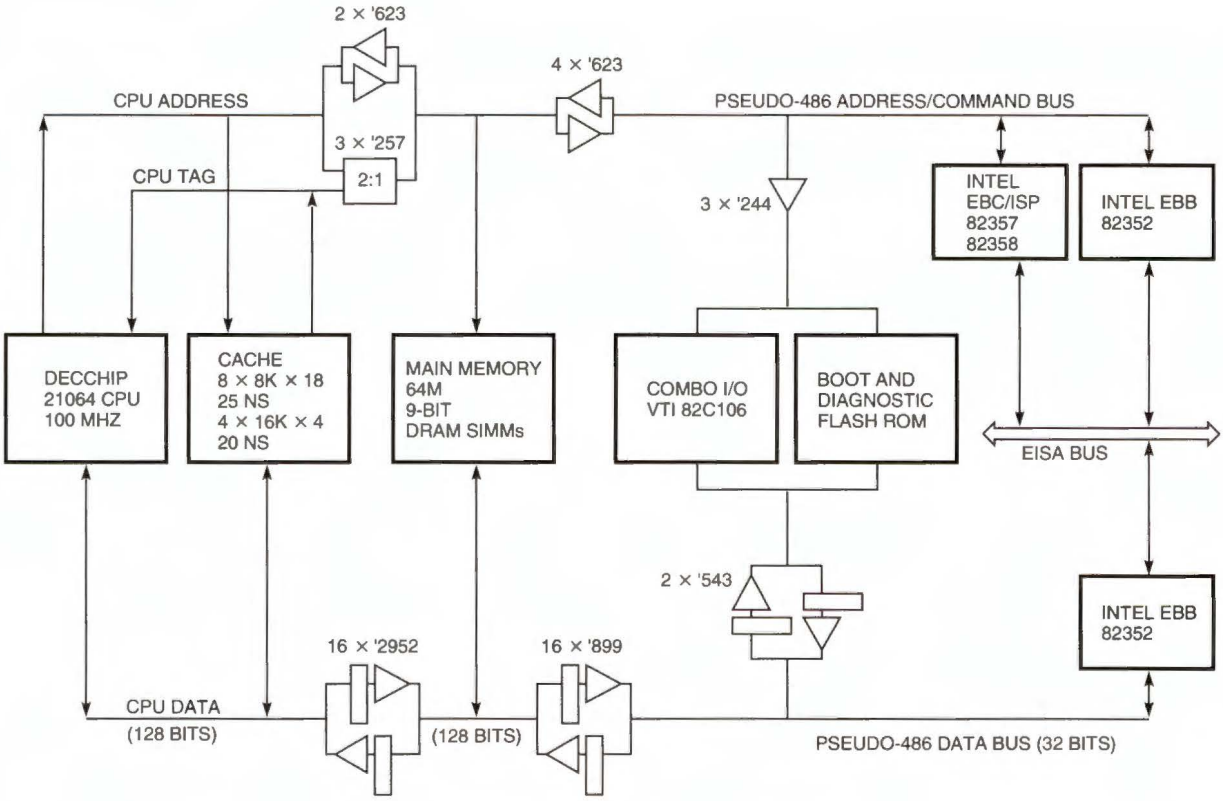


Figure 4 Block Diagram of the Theta System

knowledge gained through the Beta and Theta system development efforts in Massachusetts into this new product. The "tiger team" that was assembled in March 1992 to define and design the product gave rise to a unique partnership among development, qualification, and manufacturing groups in Hudson and Maynard, Massachusetts, and in Ayr, Scotland. The fact that the system was designed by a tiger team faced with a short, fixed schedule had a powerful effect on the architecture. The amount of time available for design work was determined by calculating backward from the time when the system was needed. The system implemented the best architecture that could be designed in the 12-week period available for design.

The schedule was extremely tight, as the following list of milestones indicates:

Date (1992)	Milestone
Mar 26	Assemble tiger team
May 29	Complete preliminary schematics
Jun 1	Begin layout
Jun 29	Complete schematics, begin parts sourcing
Jul 7	Begin prototype build
Jul 28	Complete prototype build, begin debug
Aug 5	Ship first prototype (actual date)
Aug 10	Ship first prototype (scheduled date)

The Microsoft Windows NT operating system was selected as the design center for the product. This decision made designing to an aggressive schedule even more difficult. The port of the Windows NT system to the Alpha AXP architecture was beginning at about the same time, and there were still many unknowns. This problem was solved by quickly establishing a close working relationship with key technical contributors in the Windows NT group, resulting in a design team that spanned eight time zones.

The tiger team's original intention was to base the product on the Theta design. A careful design review, however, showed that the Theta design was not well suited to high-volume manufacture. In fact, the Theta design did not implement some of the more esoteric aspects of the EISA standard, such as the asynchronous timing of translated ISA direct master cycles, and could not be easily modified to do so. Therefore, a new I/O system design was needed. This requirement caused a schedule problem for the Windows NT group. To solve the problem, the tiger team quickly defined the software-visible characteristics of the system, designed a special version of the Theta machine (which implemented this definition) and built 10 machines. The design team delivered these systems, called Theta-II, to the Windows NT group in early June 1992. The group used Theta-II systems until actual DECpc AXP 150 machines arrived in August 1992.

The DECpc AXP 150 product uses the same enclosure as the DECpc 433ST Intel-based PC. Since this enclosure was intended to hold a multiboard system, the designers' original intention was to build a multiboard system as well. To save time, however, the designers put the entire CPU, cache, memory, and core I/O system onto the mother board and placed all the complicated I/O devices (disk, network, display) on EISA option cards. Placing these devices on option cards had an additional advantage. Many system-level decisions (such as which video controller to use) were removed from the critical path and were, in fact, changed several times as the project evolved.

Because the delivery schedule was tight, the designers needed to ensure that the first-pass boards were nearly production quality (since many systems would be built using first-pass boards). The designers had electromagnetic compatibility (EMC), thermal, assembly, and test experts critique the design and incorporated as many of their suggestions as possible into the first-pass boards. Dealing with these design issues early in the schedule delayed the release of the first system boards but saved time overall because the usual flurry of changes required by regulatory testing and manufacturing was avoided.

Hardware Design

The DECpc AXP 150 hardware design is completely synchronous. The DECchip 21064 CPU chip operates at 150 MHz and generates the 25-MHz clock that runs most of the logic. Sections of the I/O system run at 8.33 MHz, generated by the EISA chip set. The original plan included a 256KB backup cache built with the 16K-deep version of the SRAMs used in the Beta and Theta systems. The designers discovered, however, that the 32K-by-9-bit SRAMs used to build caches for Intel486 DX systems were so inexpensive that building a 512KB backup cache was less expensive than building a 256KB cache. The designers reused the same basic cache design used in the Beta and Theta systems, although two copies of the cache address are needed because there are twice as many SRAMs. This design, with 17-ns SRAMs,

allows the CPU to read the cache in 32 ns and write the cache in 40 ns. Figure 5 is a block diagram of the DECpc AXP 150 product.

The memory system was redesigned to be 128 bits wide (to increase performance), to use 36-bit-wide SIMMs (to save space), to handle both 1M- and 4M-deep SIMMs, and to handle both the single- and double-banked versions of the SIMMs. Because system software strongly prefers to have contiguous memory, the design includes some hardware so that configuration software can arrange the available memory into a dense block starting at location 0.

Main memory is protected by longword parity, as in the Beta and Theta systems. The memory controller transforms partial longword DMA writes into read-modify-write cycles. The latching and merging functions are performed in the 74FCT652 transceivers situated between the memory bus and the I/O bus. The registers in the transceivers make it possible to check the parity of the old data at the same time as the new data is written, making the read-modify-write cycle faster.

The Windows NT operating system requires that a block of physically contiguous addresses on an I/O bus need only be mapped into a single block of virtually contiguous addresses in the virtual address space. Thus, schemes that place low-order Intel address bits and/or byte enables in high-order Alpha AXP address bits (like the ones used on the Beta and Theta systems) are unacceptable. The DECpc AXP 150 product, therefore, uses a new scheme, developed jointly by the hardware and software designers. This scheme places the low-order Intel address bits and the width of the cycle in low-order Alpha AXP address bits. Figure 6 illustrates the translation.

The DECchip 21066/21068 microprocessors and the DECchip 21070 chip set use a similar mapping but make two small improvements. First, they shift the Alpha AXP address 2 bits to the right, which makes the Intel addressing window 128MB in size. Second, they add a special check to make accessing the lowest 1MB of the Intel memory space more efficient, since a number of important peripherals

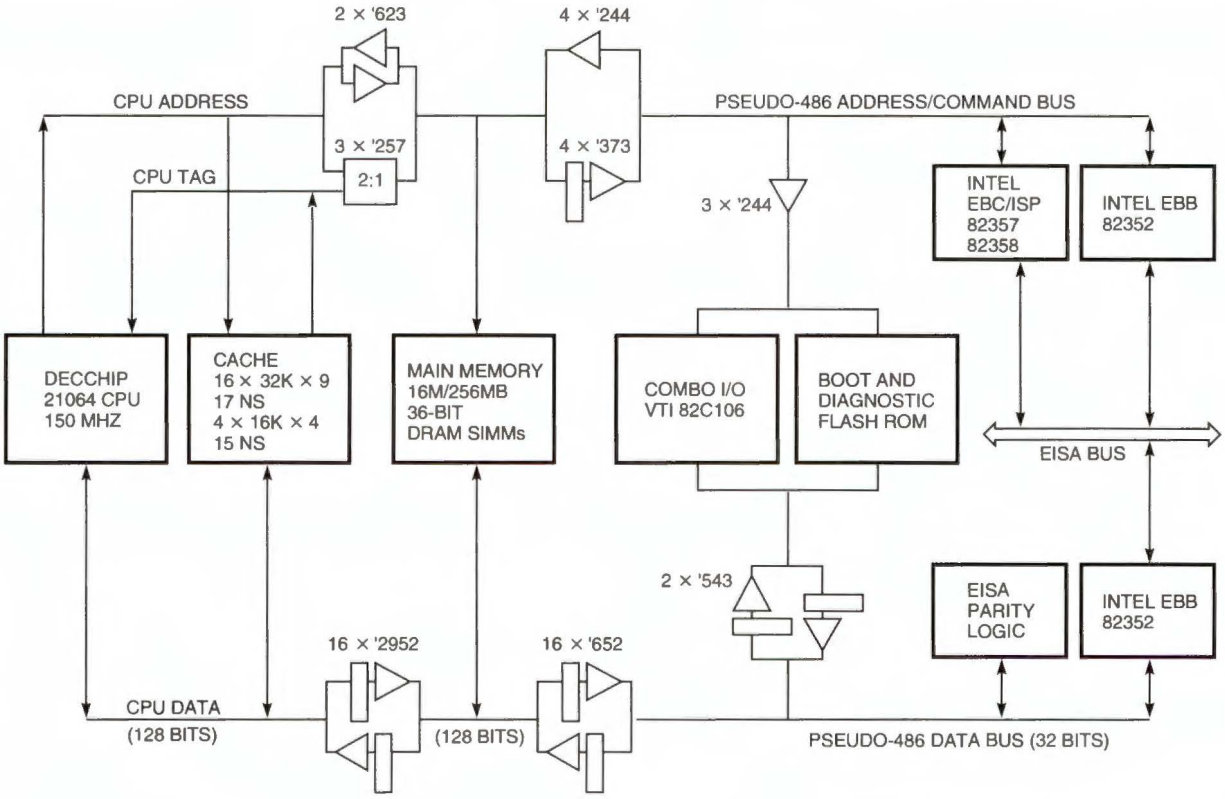


Figure 5 Block Diagram of the DECpc AXP 150 Product

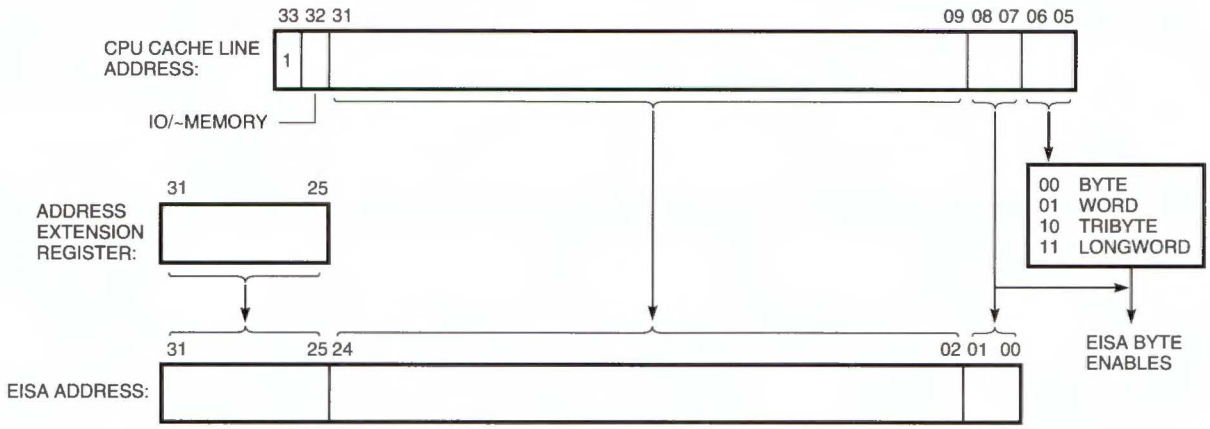


Figure 6 DECpc AXP 150 I/O Addressing

have hardwired address assignments between 640KB and 896KB.

The local I/O systems in the DECpc AXP 150 and Theta systems are similar. The only exception is that the Alpha AXP 150 system has the additional flash EPROM needed to support the console interfaces for the Windows NT operating system (thus conforming to the Advanced RISC Computing [ARC] specification) and the DEC OSF/1 AXP and OpenVMS AXP operating systems (thus conforming to the Alpha AXP console architecture). The designers considered rearranging the addressing and placing the industry-standard peripherals (those in the VLSI Technology Combo I/O chip) at their usual places in the EISA address space. This plan was rejected to ensure that the console firmware could communicate with the serial lines and be used to debug the EISA I/O system.

The EISA I/O system was tricky to design because the Intel 82350DT EISA chip set was intended to be used in a system with a very different architecture. After careful analysis of the problem, only three truly difficult issues were evident.

1. Clock skew. The EISA bus clock is generated by synchronous logic (inside the Intel 82350DT chip set) running on the system clock; however, the delays between the EISA bus clock and bus control signals, combined with the delays in the EISA bus clock generator itself, make it impossible to use EISA bus control signals as inputs to synchronous logic running on the system clock. This problem was solved by implementing the EISA control logic as two interlocked state machines. The first machine runs on the EISA bus clock, and the second machine runs on the system clock sampling the outputs of the first one. Intel uses a similar scheme in their 82350DT example designs.
2. Flow control on partial longword writes. In the DECpc AXP 150 system, the backup cache and main memory are protected by longword parity. This complicates DMA writes of less than a longword. Such writes need to be implemented as read-modify-write sequences, and the timing of the EISA byte-enable signals (which tell the DMA logic if a read-modify-write cycle is needed) make it impossible to extend the cycle using the normal EISA flow control mechanism (EXRDY). The designers solved this problem by stretching the EISA bus clock when needed; the Intel EISA chip set has logic (HSTRETCH) for doing this. The short bus clock stretch does not affect any option that conforms to the EISA bus specification.
3. ISA direct masters. The Intel 82350DT chip set contains logic that translates ISA direct master cycles into ordinary EISA cycles. These EISA cycles, however, have somewhat unusual timing; they contain events that are not synchronized to the EISA bus clock. The most difficult part of dealing with this timing was determining that these events could actually happen. Making it work required simply that some key transceiver control signals be generated combinatorially.

The arbiter in the Intel 82350DT chip set responds to bus requests (e.g., EISA bus masters or the DMA controllers inside the 82350DT chip set), stops the CPU using the DECchip 21064 CPU's holdReq/holdAck protocol, and takes control of the cache and memory system. When the CPU is in

hold, the memory controller watches for EISA memory cycles aimed at the low 256MB of memory and performs the appropriate read, write, read-modify-write, or refresh cycles. The cache cycles at the same time as memory, reading and writing as required. The timing is tighter than in previous machines, but making it work required only a careful placement of the critical parts. No attempt was made to allow the processor to run during DMA, partially to keep the design simple (EISA bursts are not required to be sequential) and partially to avoid risk. Intel systems built with the 82350DT chip set stop the CPU during DMA. Introducing parallelism could have revealed a lingering bug in the chip set that no other system had encountered.

Software Design

The designers used a custom version of the Theta-II firmware to debug the first DECpc AXP 150 systems. The systems were operational in less than half a day after they arrived from the assembly house. Almost immediately, the debugging firmware was replaced by the first version of the production firmware, developed jointly by the DECpc AXP 150 and the Windows NT firmware teams.

The first group to receive shipment of the systems was the Windows NT group, who had been using the Theta-II machines. Their software worked instantly; they only had to fix a single bug that had been concealed by a bug in the Theta-II systems. The group used both systems until enough DECpc AXP 150 systems were available, at which point the Theta-II systems were decommissioned.

Next, the OpenVMS and DEC OSF/1 groups took delivery of systems. Making each of these two systems operational on a DECpc AXP 150 machine was a slow process because this was the first time that many of the developers had dealt with PC hardware. Once operational, however, the systems stabilized rapidly.

The designers discovered few hardware problems as the operating system work progressed. Some hardware problems were discovered in the EISA option cards, when software attempted to use the cards in a manner unlike that of the MS-DOS operating system. These problems were tracked down with the help of option card vendors.

Performance

The CPU performance of the system met project expectations. The original design concept targeted 100 SPECmark89, and simulations indicated that the 150-MHz system with 512KB cache would achieve this rating. As with any new system, performance tuning is an important part of the development activity. Table 1 shows the performance results as of January 1994 for the DECpc AXP 150 system and for some of its competitors, all running industry-standard benchmarks under the Microsoft Windows NT operating system. Descriptions of these benchmarks, as well as of the hardware and software configurations used to make the measurements, can be found in the *Alpha AXP Personal Computer Performance Brief—Windows NT*.[9]

Comparing the performance of the DECpc AXP 150 system to DEC 3000 AXP workstation performance is difficult. Most performance measurements have been made under the Windows NT operating system, using the compilers and libraries appropriate for that system, and the Windows NT operating system does not run on DEC 3000 AXP systems. As shown in Table 2, the SPEC benchmark

Table 1 DECpc AXP 150 Benchmark Performance under the Windows NT Operating System

Benchmark	Metric	Digital DECpc AXP 150	Gateway 2000 P5 60 MHz	MIPS Computer Systems Magnum 75/150 MHz
Byte, numeric sort	Sorts/s	36.0	11.4	33.6
Byte, string sort	Bytes/s	136M	40.4M	123M
Byte, bit fields	Ops/s	7.6M	2.2M	7.8M
Byte, emulated float	FLOPS	2977	974	4597
Byte, simple FPU	Bytes/s	5.4M	2.7M	3.9M
Byte, transcendentals	Coeffs/s	867	334	538
Dhrystone V1.1	DMIPS	175.1	68.9	68.7
Dhrystone V2.1	DMIPS	161.5	61.6	59.8
CLinpack 100 x 100	MFLOPS	22.2	8.0	7.4 (double precision)
CWhetstone	KWIPS	95.5	34.0	36.7 (double precision)

Table 2 DECpc AXP 150 and DEC 3000 AXP Benchmark Performance under the DEC OSF/1 AXP Operating System

System	SPECint92	SPECfp92
DEC 3000 AXP Model 600	114	162
DEC 3000 AXP Model 500	84	128
DEC 3000 AXP Model 400	75	112
DECpc AXP 150	77	110 (V1.3A-4)
DEC 3000 AXP Model 300	66	92

suite, running under the DEC OSF/1 AXP operating system, demonstrates that the DECpc AXP 150 system performs like a midrange DEC 3000 AXP system. These results are not surprising because the only real difference between the two systems is the DECpc AXP 150 machine's slightly slower memory system.[10]

While adequate for many applications, the I/O performance of the DECpc AXP 150 system is limited not only by the EISA bus (which has a peak bandwidth of 33 MB/s) but also by the Intel 82350DT EISA chip set. The chip set is designed to be used with Intel microprocessors, and considerable I/O performance is lost reconciling Intel control signals with DECchip 21064 control signals. For this reason, the peak bandwidth of the EISA bus in the DECpc AXP 150 system is only 25 MB/s. The chip set also wastes EISA bus bandwidth while performing internal operations. One network adapter could transfer only 16 MB/s because of excessive EISA bus request latency introduced by the 82350DT chip set.

Outcome

The DECpc AXP 150 product was first shown in public on October 28, 1992, by Bill Gates at Windows on Wall Street, a presentation of the Windows NT operating system for more than 1,000 Wall Street analysts. The machine was subsequently shown at the Alpha AXP introduction and in both Digital's and Microsoft's booths at the 1992 Fall COMDEX conference in Las Vegas. There, the product was a finalist for "best system of show," missing the title by only one vote despite limited advertising. Digital formally announced the system at the 1993 Spring COMDEX conference, coincident with Microsoft's rollout of the Windows NT operating system. The DECpc AXP 150 product continues to receive good reviews and awards from the PC media.

Lessons Learned

Engineers involved in the Beta, Theta, and DECpc AXP 150 projects learned many lessons during the evolution of the these systems. Chief among these lessons are the following:

1. Combining Digital's Alpha AXP microprocessors and industry-standard system components was fairly straightforward and certainly within the abilities of any competent digital design engineer.
2. Even though many engineers shy away from evolutionary design because it lacks the glamour of doing things from scratch, evolution is a fine methodology. This is especially true if the cost of each step along the way is small. Each new Alpha AXP PC eliminated the obvious shortcomings of the previous designs, yet the risk of the system not working was small because most of the design was copied from the predecessor.
3. The interchip and system buses designed with Intel CPUs in mind (e.g., ISA, EISA, and Peripheral Component Interconnect [PCI]) can be used in non-Intel systems. Designing the correct interfaces, however, may require multiple iterations. Digital engineers made three iterations before arriving at what seems to be a good mapping from Alpha AXP program I/O cycles to Intel program I/O cycles.
4. Sometimes, the best way to show that something is possible is to build a demonstration unit. When the Beta system was designed, few believed that low-end Alpha AXP systems could be built. Nonbelievers found it difficult to defend their position in the presence of a working computer.
5. Clear goals can make development proceed faster, since alternatives that run counter to the goals need little analysis—they can simply be rejected. The time-to-market goal of the DECpc AXP 150 project was an extreme example. Designers needed only to consider alternatives that allowed meeting the tight schedule.
6. Qualification, test, and assembly issues must be addressed as part of the design rather than as annoying details to be addressed later. Addressing these issues during the design phase may delay the delivery of the prototype, but doing so helps to ensure that the prototype is of high quality and to avoid the risk of longer delays later in the project.

7. A good CAD system is a valuable asset. Because our CAD system was designed for rapid turnaround, it was possible to make significant changes to designs as new data appeared or when urgent needs arose. The Theta-II board design was released five days after the designers concluded that it was needed.

Acknowledgments

The contributions of many people made this effort successful. Although not complete, the following list gives credit to those who were the most deeply involved. Rich Witek, Dave Conroy, Tom Levergood, Larry Binder, Larry Stewart, Win Treese, and Joe Notarangelo designed and implemented the Beta hardware and software. Ralph Ware, Lee Ridlon, and Carey McMaster designed and implemented the Theta hardware and software. Joe Falcone first realized that there was a product lurking in the advanced development, transformed it into a real project, named it after a legendary sports car, and did the early system definition work. The Jensen hardware tiger team was led by Tom Kopec and included Jay Osmany, Fraser Murphy, Tom Mann, Dave Conroy, Lee Ridlon, and Carey McMaster. Colin Brench, Sharad Shah, Roy Batchelder, Jim Pazik, and Mike Allen provided consulting services early in the design.

Note and References

1. The DEC 2000 AXP Model 300 is the same product using the DEC OSF/1 AXP and the OpenVMS AXP operating systems.

2. R. Sites, ed., *Alpha Architecture Reference Manual* (Burlington, MA: Digital Press, Order No. EY-L520E-DP, 1992).

3. *DECchip 21064 Microprocessor: Hardware Reference Manual* (Maynard, MA: Digital Equipment Corporation, Order No. EC-N0079-72, 1992). This document and many others associated with Digital's Alpha AXP microprocessors are available on the Internet and can be obtained via anonymous FTP from gatekeeper.pa.dec.com, in directory pub/DEC/Alpha.

4. H. Johnson and M. Graham, Chapter 6 in *High Speed Digital Design: A Handbook of Black Magic* (Englewood Cliffs, NJ: Prentice-Hall, 1993).

5. E. Solari, *ISA and EISA, Theory and Operation* (San Diego, CA: Annabooks, 1992).

6. C. Thacker, D. Conroy, and L. Stewart, "The Alpha Demonstration Unit: A High-Performance Multiprocessor," *Communications of the ACM,* vol. 36, no. 2 (February 1993): 55–67.

7. J. Gettys, personal communication, 1991.

8. *82350DT EISA Chip Set* (Santa Clara, CA: Intel Corporation, Order No. 290377-003, 1992).

9. *Alpha AXP Personal Computer Performance Brief—Windows NT,* 2d ed. (Maynard, MA: Digital Equipment Corporation, Order No. EC-N2685-10, January 1994).

10. *Alpha AXP Workstation Family Performance Brief—DEC OSF/1 AXP,* 4th ed. (Maynard, MA: Digital Equipment Corporation, Order No. EC-N2922-10, February 1994).

Dina L. McKinney
Masooma Bhaiwala
Kwong-Tak A. Chui
Christopher L. Houghton
James R. Mullens
Daniel L. Leibholz
Sanjay J. Patel
Delvan A. Ramey
Mark B. Rosenbluth

Digital's DECchip 21066: The First Cost-focused Alpha AXP Chip

The DECchip 21066 microprocessor is the first Alpha AXP microprocessor to target cost-focused system applications and the second in a family of chips to implement the Alpha AXP architecture. The chip is a 0.675-micrometer (μm), CMOS-based, superscalar, superpipelined processor that uses dual instruction issue. It incorporates a high level of system integration to provide best-in-class system performance for low-cost system applications. The DECchip 21066 microprocessor integrates on-chip, fully pipelined, integer and floating-point processors, a high-bandwidth memory controller, an industry-standard PCI I/O controller, graphics-assisting hardware, internal instruction and data caches, and an external cache controller. Cost-saving packaging techniques and an on-chip, analog phase-locked loop enable the chip to meet the cost demands of personal computers and desktop systems. This paper discusses the trade-offs and results of the design, verification, and implementation of the DECchip 21066 microprocessor.

The DECchip 21066 microprocessor is the first cost-focused implementation of the Alpha AXP architecture.[1] The chip integrates system functions that are normally found in a microprocessor chip set with a high-performance, superscalar microprocessor core to deliver high-end personal computer (PC) performance and low overall system cost. The DECchip 21066 device is also the first microprocessor to integrate a Peripheral Component Interconnect (PCI) local bus controller.[2] This open industry-standard I/O bus allows direct connection to high-performance peripheral components supplied by many vendors. The bus also allows direct connection to commodity Industry Standard Architecture (ISA) or Extended ISA (EISA)-based PC peripherals through a simple bridge chip. The DECchip 21066 microprocessor integrates a high-bandwidth memory controller that directly sequences an external secondary cache and main system memory. In addition, an on-chip, phase-locked loop (PLL) multiplies a low-frequency reference clock to a high-frequency CPU core clock, thus eliminating the need for a high-frequency board oscillator. The combination of cost-saving functional integration and features that ease system design reduces the overall CPU subsystem cost and shortens the time-to-market of high-volume Alpha AXP PCs.

CMOS Technology

The first chip to fully implement the Alpha AXP architecture, the DECchip 21064 microprocessor was designed in a 0.75-micrometer (μm) (drawn) complementary metal-oxide semiconductor (CMOS) process.[3,4] The physical implementation of the DECchip 21066 microprocessor was achieved through the use of a 10 percent CMOS process shrink, which reduced the minimum feature size from 0.75 μm (drawn) to 0.675 μm. This reduction in the feature size enabled the DECchip 21064 integer unit, floating-point unit, and caches to be combined with a new memory controller, PCI I/O controller, and PLL on a die with approximately the same area as the original DECchip 21064 device. The maximum core clock speed of the DECchip 21066 microprocessor is specified as 166 megahertz (MHz).

This speed allows the internal PLL to multiply a 33-MHz reference clock (the same reference clock as on the PCI bus) by five to generate a clock for the CPU core. This cycle time was set to be slightly less aggressive than the 200-MHz maximum core clock speed for the DECchip 21064 microprocessor to allow a greater number of yielding parts and thus lower part cost.

Target Market

One target market for the DECchip 21066 microprocessor is Alpha AXP PCs running the Microsoft Windows NT operating system. The bulk of the application base for this operating system is provided by vendors who target the Microsoft Windows operating system running on PCs based on Intelx86 microprocessors. Software vendors have substantial expertise with this architecture and, until recently, the advanced software tool base for Windows development has been targeted exclusively at the Intelx86 architecture. To provide compelling motivation for application developers to supply portable Windows NT applications and for customers to adopt a new architecture, Alpha AXP PCs must be price competitive with Intelx86 PCs and must deliver substantially higher performance.

The availability of the Windows NT operating system and high-performance PCs based on Intel's Pentium microprocessor dictated that a set of competitive products be available in early 1994.[5] Furthermore, the performance of Pentium-based PCs set the lower bound of acceptable performance for PCs based on the Alpha AXP technology. These schedule and performance goals were met by implementing the DECchip 21066 microprocessor as a high-integration design variant of the DECchip 21064 microprocessor. This strategy permitted a design cycle of only 10 months.

Target Performance

The DECchip 21066 microprocessor provides a CPU subsystem that is comparable in cost to a 66-MHz Intel486 DX2-based PC with Pentium-class performance.[6] Because of its excellent price/performance ratio, the DECchip 21066 microprocessor is attractive to low-end workstation products running the DEC OSF/1 AXP and OpenVMS AXP operating systems, in addition to Windows NT platforms.

A performance-limiting factor for many applications is the bandwidth to the second-level cache. The DECchip 21064 microprocessor maximizes bandwidth with a 128-bit data bus, which allows two secondary cache read sequences to fill a line in the primary cache. DECchip 21066 engineers chose to implement a 64-bit data bus. This implementation resulted in a less-expensive package, a smaller die, and lower system cost, while reducing by only 20 percent the performance that would have been achievable with a 128-bit bus. The fast core clock and close proximity of the secondary cache to the core still enable the DECchip 21066 microprocessor to deliver high performance.

Internal Components

The block diagram in Figure 1 shows the major DECchip 21066 microprocessor components. Fetched instructions from an on-chip 8-kilobyte, direct-mapped instruction cache are dual-issued to the integer, floating-point, and load-and-store (addressing box) units. The on-chip, 8-kilobyte, direct-mapped data cache initially services memory loads. Stores are written through the data cache and absorbed into a write buffer. The memory controller or PCI I/O controller handles all references that pass through the first level of caches.

The memory controller handles memory-bound traffic. The controller first probes a direct-mapped, write-back, write-allocate cache and then sequences main memory to fill the caches, if necessary. The PCI I/O controller handles I/O-bound traffic and performs programmed I/O read and write operations on behalf of the CPU. Direct memory access (DMA) traffic from the PCI is handled by the PCI controller in concert with the memory controller. DMA read and write operations are not allocated in the secondary cache. The memory and PCI interfaces were designed specifically for uniprocessor systems and do not support multiprocessor implementations.

System Application

Figure 2 shows a sample system block diagram using the DECchip 21066 microprocessor. In this configuration, the memory controller sequences both the static random-access memory (SRAM) secondary cache and the dynamic random-access memory (DRAM) main memory. The secondary cache comprises tag and data SRAMs with identical read and write access times. System software selects specific random-access memory (RAM) timing values in increments of the number of core clock cycles.

The design supports four separate banks of DRAM, each of which can be timed independently. This feature adds flexibility in memory organization and upgrading. One of the four banks can be configured

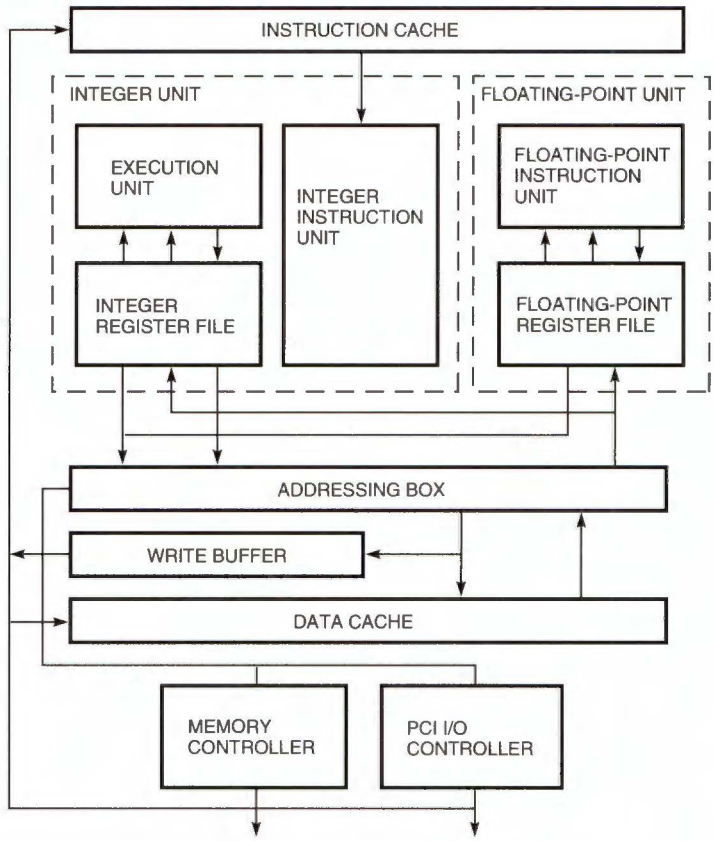


Figure 1 Major Components of the DECchip 21066 Microprocessor

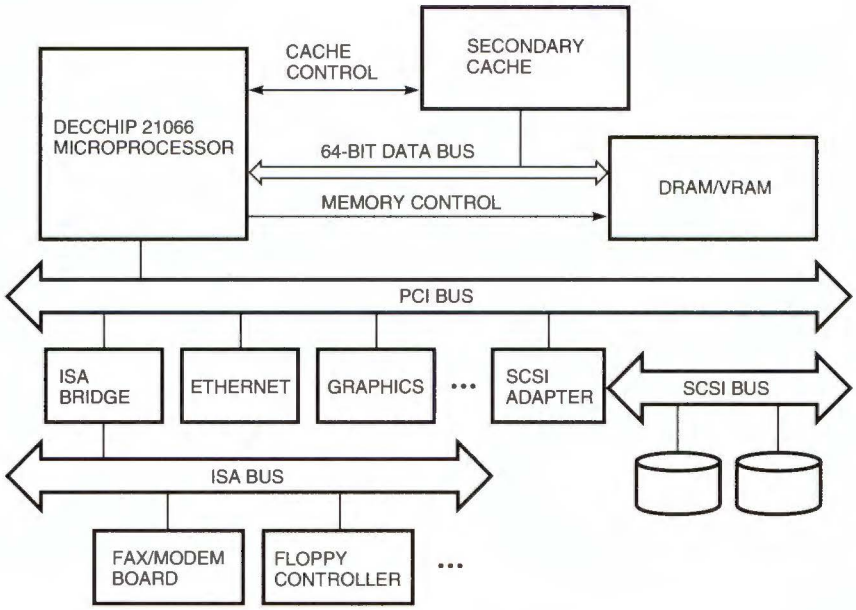


Figure 2 Sample System Block Diagram

with video RAMs (VRAMs) for low-cost graphics applications. The memory controller directly sequences the VRAMs and supports simple graphics operations, such as stippling and bit-focused writes.

Even though the DECchip 21066 microprocessor operates at 3.3 volts (V), all pads directly drive transistor-transistor logic (TTL) levels and can be driven by either 3.3-V or 5-V components. This eliminates the need for either special voltage conversion buffers or 3.3-V memory parts, which can add to system cost or affect performance.

The PCI bus is a high-bandwidth bus with a number of attractive features. In addition to its ability to handle DMA and programmed I/O, the PCI bus allows for special configuration cycles, extendibility to 64 bits, 3.3-V or 5-V components, and faster timing. The base implementation of the PCI bus supports 32 bits of multiplexed address and data with a bus clock of 33 MHz, yielding a maximum burst bandwidth of 132 megabytes per second (MB/s).

The PCI bus is directly driven by the microprocessor. In Figure 2, some high-speed peripheral devices, such as graphics and small computer systems interface (SCSI) adapters, connect directly to the PCI bus. An ISA bridge chip allows access to lower-cost, slower peripheral devices, such as modems and floppy disk drives.

Memory Controller Features

A primary goal of the DECchip 21066 project was to simplify the system design to enable customers to build products with minimal engineering investment. Consequently, the memory controller directly connects to industry-standard DRAMs and single in-line memory modules (SIMMs), with buffers required only for electrical drive. To span the wide range of system applications from embedded controllers to midrange workstations, the timing of address, data, and control signals is highly programmable. This feature supports varying speeds and physical organization of the DRAMs and clock speeds of the DECchip 21066 microprocessor itself.

The memory controller supports from one to four banks of memory, which allows main memory to range from 2M bytes to 512M bytes. The system may include a secondary cache of 64K bytes to 2M bytes. This optional cache also connects directly to the microprocessor, and the timing of control signals is programmable in increments of chip cycles.

Several simple, graphics-assisting features allow systems that need to save the cost of dedicated graphics control hardware to control the system frame buffer in software. The decision to include graphics features was based on the ability to provide significant acceleration of key functions for minimal hardware cost in the memory controller.

Dynamic Memory Interface

The DRAM data interface consists of a 64-bit-wide bidirectional bus that is shared with the secondary cache. The interface provides two control signals for an optional memory data bus transceiver that may be required because of bus loading. All main memory read operations return a full 64 bits of data. Main memory write operations normally consist of a full quadword (64-bit) write. Write requests for less than a quadword of data automatically result in a read-modify-write sequence, unless the software has enabled masked writes on the memory bank. During masked writes, the error correction code (ECC) pins provide a byte mask that is externally combined with a column address strobe (CAS) signal to form a byte mask strobe for the array. ECC checking is not supported on memory banks that have masked writes enabled.

An optional 8-bit bidirectional bus provides quadword ECC checking on the main memory and the secondary cache. Each bank of main memory may be protected by ECC at the cost of including 8 extra DRAM bits per bank. The ECC used by the DECchip 21066 microprocessor allows correction of single-bit errors and detects double-bit and 4-bit nibble errors. The memory controller automatically corrects the single-bit memory read errors before data is passed to the read requester. When the ECC identifies a double-bit or a 4-bit nibble error, the memory controller does not attempt to correct it. When any ECC error is detected, the memory controller stores the error condition along with the address of the error. System software must scrub the error from physical memory.

Secondary Cache

The DECchip 21066 microprocessor supports a secondary cache designed with industry-standard asynchronous SRAMs. The cache is direct mapped with 8-byte blocks and uses a write-back policy. Designers chose a secondary cache block size that is smaller than that of the 32-byte primary internal caches to simplify the allocation of an external cache block during write operations. With the smaller block size, it is not necessary to fetch the missing quadwords of the block from DRAM when less than a full internal cache block is written, as

would be the case with a larger block size. In addition, the smaller block size provides one dirty bit per quadword, so that the resolution of dirty (not yet written back) data is one quadword rather than four, thus reducing the number of write-back operations to the DRAM.

The DECchip 21066 microprocessor writes the cache tag into the SRAMs on cache allocation operations and receives and compares the tag internally on cache lookup operations. Read-fill operations of the internal caches are pipelined as the microprocessor drives the next cache index to the SRAMs prior to determining the hit or miss result of the current lookup. A cache hit, which is the more common occurrence, causes SRAM access to achieve maximum speed. In the case of a miss, the microprocessor re-drives the miss address and reads the DRAMs.

The chip divides the memory space into cacheable and noncacheable regions based on address. To save time, accesses to the noncacheable region skip the cache-probe step and immediately access the DRAM. This feature may be used, for example, to optimize accesses to a frame buffer, which typically is not cached.

The cache tag field, including the dirty bit, may optionally be protected by a single parity bit. In systems with write-through caches, tag parity errors are not necessarily fatal since the correct data can be fetched from DRAM. With a write-back external cache, such as the one used by the DECchip 21066 microprocessor, the bad parity may be on a dirty location. In this case, bad parity is a fatal error, since the copy in DRAM is no longer current.

Graphics-assisting Functions

The graphics-assisting logic of the DECchip 21066 microprocessor provides some basic hardware enhancements to improve frame buffer performance over a standard, simple frame buffer system. The graphics-data-path–assist logic is targeted at reducing the number of instructions executed during inner loop graphics operations. By reducing the number of inner loop instructions, the microprocessor offers an improved graphics performance while keeping the overall graphics subsystem cost in line by using a simple frame buffer design rather than a more expensive graphics accelerator. The graphics-assist logic provides the opportunity to design a low-cost, entry-level graphics option and may be used in conjunction with a high-performance graphics accelerator.

The hardware graphics-assist logic consists of a direct interface to VRAM parts and the ability to perform some simple graphics-oriented data manipulations. To facilitate the use of VRAMs, the memory controller supports both full- and split-shift register load operations. External video monitor control logic generates monitor timing and signals the DECchip 21066 memory controller when VRAM shift register loads are required. An internal linear address generator keeps track of the video display's refresh address.

The features of the graphics data path provide the ability to perform simple frame buffer–style data operations, e.g., transparent stipple writes, plane masked writes, and byte writes (with minimal external logic). In transparent stipple mode, the memory controller can conditionally substitute the foreground data for the frame buffer data on memory writes.

Pixel depth may range from 1 bit to 32 bits. A special feature of the graphics-assist logic is the ability to perform graphics data manipulation operations on both VRAM and standard DRAM memory banks. The DECchip microprocessor emulates graphics operations targeted at memory banks without VRAMs by means of a read-modify-write sequence. This method allows the same graphics firmware to operate on either VRAM or DRAM memory banks and allows DRAM to be used as additional off-screen memory.

A low-cost video option board may be built around the microprocessor by adding a bank of VRAMs and a video controller. Typical video controller logic consists of a video timing generator, a RAMDAC (video digital-to-analog [D/A] converter with color mapping), a hardware cursor, and a few other simple video data path components.

PCI Bus Interface

The DECchip 21066 microprocessor is the industry's first microprocessor to implement an interface that connects directly to the PCI bus. This PCI bus interface, called the I/O control (IOC), runs asynchronously with the rest of the microprocessor's core logic. Asynchronous design was chosen to enable optimal system performance by setting the chip's core clock to its maximum frequency without being limited to an integer multiple of the PCI clock frequency.

The IOC can be viewed as two separate controllers: one for DMA and the other for core requests. The DMA controller handles all peripheral-initiated

DMA operations to the system memory; the core controller handles the loads and stores to either the PCI devices or the IOC internal registers. Since the PCI bus uses a 32-bit address and the DECchip 21066 microprocessor uses a 34-bit address, a PCI address to which the IOC responds must be translated to an equivalent address in the microprocessor's address space. The IOC provides two types of address-translation mechanisms, direct and indirect (also called scatter-gather).

During a direct-mapped address translation, the lower bits of the PCI address are concatenated with a page address that is stored in an IOC register to form a 34-bit translated address. In an indirect-mapped address translation, certain bits of the PCI address are concatenated with a translated base address that is stored in an IOC register to form a 34-bit address. This address indexes into a scatter-gather map table in system memory where the page address resides. The page address is then concatenated with the lower bits of the PCI address to form the translated address. The IOC contains two programmable windows, each of which can be programmed to respond with direct or indirect address translation. To facilitate fast translation for the indirect address translation, the IOC contains an eight-entry, fully associative, translation look-aside buffer. To simplify the design, DMA burst length is limited to occur within an 8K-byte page boundary. Bursts that extend beyond a page boundary are broken into separate transfers.

CPU addresses are translated to an equivalent address in the PCI address space through one of two types of address translation, sparse or dense, depending on the target region of the address. For sparse-space access, the lower 32 bits of the CPU address are shifted right by 5 bits to generate a 27-bit address. This address, concatenated with a 5-bit field from a register in the IOC, maps the 32-bit PCI address. Transfers of up to 8 bytes can be completed in this address space. For dense-space addressing, the lower 32 bits of the CPU address are directly mapped to the PCI address. Only unmasked operations are allowed, and up to 32 bytes of write data and 8 bytes of read data can be transferred in this address space. The IOC improves write bandwidth by buffering two 32-byte writes from the DECchip 21066 core.

The priorities of the IOC design were to support peak PCI bandwidth and at the same time to meet the tight project schedule and limited die area constraint. A 32-byte data queue buffers DMA data. A larger queue would require more die area; a smaller queue might stall the DMA bursts due to the synchronization delay and system memory access latency. The IOC and memory controller are connected by a 64-bit data bus. Since the IOC PCI bus supports only 32 bits, two 32-bit data words are transferred at a time to minimize read-modify-write operations that would otherwise be issued by the memory controller. The IOC improves the DMA read bandwidth by prefetching up to 32 bytes of data.

A hardware semaphore provides the asynchronous handshaking between the core clock and PCI clock domains. Approximately two months were dedicated to the logic and physical implementation of the hardware semaphore and the manual verification of the asynchronous communication and signal timing. This effort was required because the existing computer-aided design (CAD) tools did not have a formal verification methodology that would sufficiently validate the asynchronous aspects of the IOC architectural design and the physical implementation.

Developing the test strategy was a challenge for two reasons: (1) the presence of asynchronous sections and (2) the need to guarantee that the semaphore would yield predictable results on a production tester operating at the maximum design frequency (166 MHz for the core and 33 MHz for the PCI bus). Even when the two clocks are running synchronously, they must be controlled and aligned such that the hardware semaphore guarantees reliable and predictable results. Each core clock phase is only 3 nanoseconds at 166 MHz. Taking into account the logic delay and setup time required by the hardware semaphore, this clock skew requirement is essentially impossible to achieve. The inaccuracy of the tester and the test hardware further complicates the problem. The solution adopted was to incorporate extra logic to reduce the sample rate of the hardware semaphore that runs on the core clock without actually slowing down the clock itself. With the extra logic, we were able to demonstrate the ability to test the IOC and the DECchip 21066 microprocessor at the targeted clock frequencies.

Logic Design Verification Process

A pseudorandom design exerciser is the main verification tool for the DECchip 21066 microprocessor. The exerciser takes input from Alpha AXP code streams and PCI bus commands. A template-based text manipulator generates test patterns from the

templates written by verification engineers to target specific sections of the microprocessor design. The text manipulator provides the primary source of randomness. It selects templates based on probabilities to create the test pattern. Each test pattern runs on a design model. We use two different design models, a register transfer level (RTL) model written in the C programming language and a structural model generated from the circuit schematics. Finally, the same test pattern runs on the instruction set processor (ISP) level reference model to determine the correctness of the design model.[7] Figure 3 illustrates the logic design exerciser process flow.

To determine if the design model operates correctly, the exerciser performs checks at three points: (1) during the execution of the design model, (2) during the execution of the reference model, and (3) after both models complete. Verification software in the design model performs additional checks for illegal conditions, such as multiple bus drivers, protocol violations, and invalid states. This code is kept to a minimum and covers design aspects not easily checked by comparing state information.

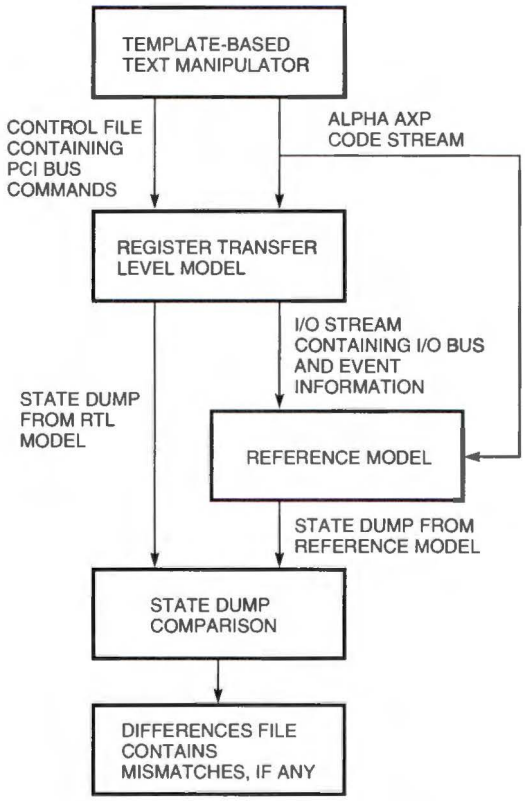


Figure 3 Logic Design Exerciser Process

Previous verification projects have used similar random test methods; however, they did not have to ensure correct operation of asynchronous I/O transactions.[7,8] To deal with this problem, the DECchip 21066 verification team enhanced the reference model to evaluate CPU and PCI transactions. The reference model receives the same stimuli as the design model plus additional I/O bus and event information. While executing the test pattern, the model verifies that the I/O information is consistent.

By periodically comparing functional states in a test pattern, we circumvented adding detailed timing information to the reference model. The model needs only valid state for comparison at synchronized capture points rather than at a clock phase boundary. In essence, we traded off being able to determine the timing correctness of our design for a simpler, faster reference model. We augmented the design model with verification code to check critical timings and performance features. Detailed static and circuit timing analysis was accomplished by other CAD tools.

Shared Memory

Several features of the DECchip 21066 microprocessor require the exercisers to employ a shared memory structure to properly test the chip.

- The microprocessor invalidates an internal data cache line on a DMA transfer to that cache line address.
- An exclusive PCI access unconditionally clears the processor's lock flag. A nonexclusive DMA write also clears the lock flag if the DMA address is within the locked 32-byte range.
- The DECchip 21066 secondary cache does not allocate on a DMA operation. Hence, the only way to test DMA sequences that hit in the secondary cache is to first access those addresses from the CPU.

The prime challenge to implementing a shared memory is that simultaneous accesses to the same address location by the CPU and DMA could result in unpredictable behavior. To circumvent this problem, we developed a model for software access to shared memory and added synchronization hooks into the reference model and the I/O trace file. The shared memory access model and synchronization hooks extended the exerciser to fully verify the DECchip 21066 shared memory capability.

The shared memory implementation proved useful in discovering two bugs in the shared memory functionality. Both bugs were tricky to generate and would have been extremely difficult to find without the shared memory capabilities of a random exerciser.

The random verification methodology enabled us to meet the aggressive 10-month schedule. By separating functional correctness and timing correctness, we were able to quickly implement and stabilize the exerciser environment. The exerciser can easily be ported to future generations of the DECchip 21066 microprocessor; we have already used the tool to verify the second-pass design.

PLL Design Issues

To reduce the module-level cost of a system based on the DECchip 21066 microprocessor, the chip includes a frequency synthesis system based on a phase-locked loop design.[9-18] The module frequency reference is a standard crystal oscillator, rather than an expensive, surface acoustic wave (SAW) oscillator. This design reduces radio frequency emissions, making qualification of the design easier and reducing system enclosure cost. Different speed binnings of the chip can run at maximum performance levels in the same basic module design. This result is possible because of the asynchronous, fixed clock rate of the PCI bus, the on-chip caches, and the programmable timing of the memory interface.

Incorporating a PLL subsystem in a digital CPU chip design requires that several problems be addressed. For example, increased noise sensitivity, which is a concern for digital designs, is even more critical for the analog circuits in a PLL. Often, simple guidelines for digital designs can ensure reasonable noise immunity; however, analog circuits frequently require detailed SPICE simulations that incorporate package- and chip-switching noise models.[19] Designers must consider making changes to otherwise fully functional circuits to ensure noise immunity. These changes can range from appropriate decoupling of critical nodes with high-quality, on-chip capacitors to consideration of major circuit changes or additions. The overall solution must be some combination of the following three alternatives: (1) careful circuit design based on noise environment simulations, (2) decoupling that is sufficient to reduce noise to tolerable levels, and (3) a better package that reduces loop inductances to reduce switching noise on the internal power supplies.

Each of the three choices comes with an associated cost or risk. The cost benefits of having a PLL at the system level must be sufficient to justify the extra design time, chip area, and risk of excess jitter and low yields, which increase chip cost. On the first Alpha AXP microprocessor project, these factors were not critical because high-end systems can afford the extra module and system cost to reduce the risk associated with delivering the product to market. With the lower-cost targets of the DECchip 21066 project, eliminating expensive SAW oscillators in favor of less-expensive, standard crystal devices is critical and justifies the chip-level costs and risks associated with introducing a new analog subsystem into the chip design.

Figure 4 illustrates how the PLL is a closed-loop feedback system. An output is compared with an input to determine the degree of error. The error signal is then used to make adjustments within the elements of the system to reduce the error. This process occurs continually; under normal operating conditions, the error approaches zero. The major components of the PLL are

- A phase frequency detector and a charge pump. The phase frequency detector inputs the reference and feedback signals and measures the phase error. The charge pump outputs a current proportional to this error, using digital pulse-width modulation to achieve an analog signal.
- A filter. A simple, low-pass, resistor-capacitor (RC) filter averages the error signal, removing the digital noise of the pulse-width modulation used by the charge pump and setting the response rate of the PLL while ensuring loop stability. The filter is composed of an on-chip, 100-ohm resistor and the external capacitor.
- A voltage-controlled oscillator (VCO). The VCO consists of a voltage-to-current buffer and a current-controlled oscillator. The output frequency of the VCO is proportional to the input voltage.
- A feedback frequency divider (K_NDIV). In the linear control theory model of the PLL, the feedback signal is the oscillator phase attenuated by the gain $1/N$, where N equals the number of clock pulses input for every output pulse. This produces an oscillator phase output that is N times the input, which is the key to synthesizing a higher-frequency clock from a lower-frequency reference. A digital divider reduces the frequency of the VCO, thus reducing its

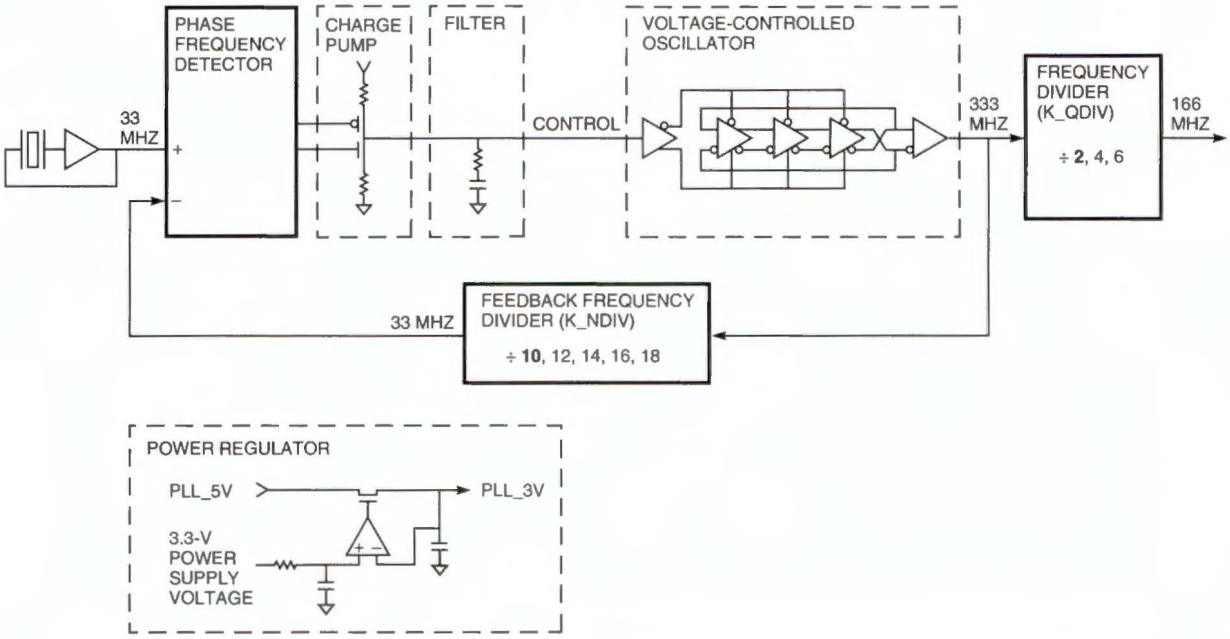


Figure 4 Overview of the DECchip 21066 Phased-locked Loop Frequency Synthesis System

phase as well. In this manner, a digital divide by *N* performs an analog attenuate-by-*N* function.

Although not strictly part of the PLL itself, the following two design elements are major components of the frequency synthesis system:

- Power regulators. To reduce clock jitter, the PLL on-chip power is regulated using the noisy module 5-V power input. The use of power regulators yields a power source with less AC noise than the global 3.3-V power used by the digital logic and I/O pads.
- A simple, programmable frequency divider (K_QDIV). This divider reduces the VCO frequency while ensuring a 50 percent duty cycle. The K_QDIV output is then buffered adequately to drive the global clock node.

The RC filter and VCO create a textbook second-order feedback loop. This model uses linear control theory methods to analyze loop stability. The on-chip resistor is a key element in loop stability because it creates the transfer function zero that ultimately ensures stability. The off-chip capacitor, however, subjects the PLL to chip-switching noise coupled between the signal etches of the package. To minimize the effects of this AC noise on clock jitter, the design includes additional decoupling capacitors, such as the one shown in Figure 5. The capacitors add additional high-frequency poles. These poles are placed well above the PLL bandwidth, and are thus largely ignored when analyzing stability, but well below the package resonance frequencies that would cause significant jitter.

The implementation of the oscillator can take many different forms. The PLL design described in this paper uses a ring oscillator based on simple differential buffers. The reasons for this design decision are as follows:

- Power supply current is nominally constant because the summation of current into each buffer in the ring smoothes the total.
- Differential buffers used in the ring offer potential power supply rejection ration (PSRR) benefits.
- Multiple phase outputs are possible. Although the DECchip 21066 microprocessor does not use multiple phase outputs, other applications of this basic oscillator design have exploited the feature.
- Designers' experience with this oscillator implementation minimizes the risk, as compared to possible multivibrator designs.

The current design provides only for setting the clock frequency synthesis ratio at power up. Future designs may include the ability to control the

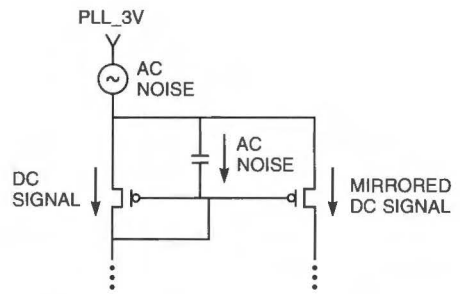


Figure 5 Use of a Decoupling Capacitor to Reduce Noise in an Analog Current Mirror

synthesis ratio through software. Such control can provide a means of lowering chip power levels as well as more flexibility over using different clock rates in the same basic module design.

The off-chip capacitor in the low-pass loop filter does have an associated cost. Package coupling introduces noise on the control voltage, so placement of this capacitor is important. Future designs will likely use on-chip filters to reduce this constraint.

The power regulators are effective, but using 5 V will not be possible as CMOS technologies move to shorter channel lengths. The trade-offs between on-chip regulation and different oscillator buffer designs that have better jitter characteristics for noisy power levels are yet to be investigated.

Package Development

The DECchip 21066 microprocessor is the first CMOS microprocessor with a cost-focused package. Designers followed tight cost constraints to allow the package to be used on follow-on chip variants for the embedded microprocessor market. With a goal of 200-MHz operation, significant electrical analysis was necessary to ensure adequate package performance.

We learned from previous designs that the main concern regarding package design is ensuring the integrity of the on-chip supply voltage. For a given chip design and performance goal (which determines power dissipation, specifically, the chip's supply current characteristics), primarily one package characteristic and one chip characteristic influence on-chip supply integrity. These characteristics are package supply loop inductance (L_l), which is the intrinsic parasitic inductance of the supply path through the chip package, and on-chip supply decoupling capacitance (C_d). The on-chip supply integrity is generally proportional to the square root of C_d/L_l.

The supply loop inductance affects on-chip supply integrity because variations in the chip's supply current generate inductive noise voltage on the on-chip supply voltage. The magnitude of supply current variations is dependent on chip clock frequency. At 200 MHz, variations in supply current can be as much 3 amperes (A). If not reduced by some method, the inductive noise generated across the supply loop inductance would be excessive.

Three methods are typically used in microprocessor chip and package designs to decrease supply loop inductance: (1) increase the number of package supply pins, (2) increase the number of internal package layers (sometimes called planes), and (3) include on-package decoupling capacitors to shunt the supply loop inductance component of the package pins.

Given DECchip 21066 package cost constraints, designers ruled out several package features used in previous designs, notably the DECchip 21064 design. For example, on-package decoupling capacitors were considered too expensive, since they would have added approximately 50 percent to the package cost. To limit the package size, the number of available package pins was constrained to 287. Specifically, the number of package supply pins was reduced to 66 percent of the number implemented on the DECchip 21064 microprocessor. The number of internal package supply planes was reduced from four to two. Looser tolerances on internal package geometry allow the DECchip 21066 packages to be manufactured in the package vendors' less-costly process lines; however, less-than-optimal supply routing results.

Given the constraints and the lack of complete package simulation models, we developed a method that combined simulation and empirical analysis to ensure adequate on-chip supply integrity. The expected package supply loop inductance was determined through a two-step process:

1. We modified the packages of a set of DECchip 21064 parts (by removing the on-package decoupling capacitor and reducing the number of supply pins) to closely match the proposed package configuration for the target chip. We determined the L_l of the degraded DECchip 21064 packages from the following relationship derived from the classical formula for the resonance frequency of an inductive and capacitive tank circuit:[20]

$$L_l = \frac{1}{4\pi^2 F_r^2 C_d},$$

where F_r is the measured supply resonance frequency of the DECchip 21064 supply network, and C_d is the known on-chip supply decoupling capacitance for the DECchip 21064 microprocessor.

2. We used simulation to determine the effect on the supply loop inductance of eliminating expensive internal package features.

The resulting expected supply loop inductance for the DECchip 21066 packages was factored into the following relationship, which must be satisfied for the on-chip supply integrity (on-chip supply noise voltage) of the DECchip 21066 microprocessor to be equivalent to that of the DECchip 21064 device:

$$\frac{\sqrt{\frac{C_{d\_21064}}{L_{l\_21064}}}}{S_{21064}} = \frac{\sqrt{\frac{C_{d\_21066}}{L_{l\_21066}}}}{S_{21066}}$$

$$= Q_{21064}$$

= quality factor of the DECchip 21064 on-chip supply,

where S is the maximum difference in chip supply current (I_s). For example, the DECchip 21064 microprocessor operates with a minimum I_s of 6 A and a maximum I_s of 9 A (both at 200 MHz). Therefore, S_{21064} is equal to 3 A.

The amount of on-chip decoupling capacitance specified by this relationship was impractical to implement in the DECchip 21066 design. Because achieving the on-chip supply integrity of the DECchip 21064 design was not possible, we investigated the extent of supply integrity degradation that could accompany a reliably functioning chip. We learned from system tests that DECchip 21064 chips with a quality factor as low as one-third the quality factor of original DECchip 21064 chips function correctly up to a chip frequency of 135 MHz.

Because the system testing was not exhaustive and because this testing could not be performed above 135 MHz, we wanted to further increase our quality factor. We added on-chip decoupling capacitance in areas not otherwise utilized and implemented package features that would minimize the supply loop inductance. Consequently, the DECchip 21066 microprocessor ultimately achieved a quality factor equal to 65 percent that of the DECchip 21064 microprocessor.

As a result of system testing the DECchip 21064 at reduced quality factors, Q_{21066} equal to 65 percent of Q_{21064} was deemed adequate. The recent opportunity to test and measure DECchip 21066 parts validated the decision to reduce the quality factor to this level.

Summary

This paper describes the DECchip 21066 microprocessor, the first cost-focused chip to implement the Alpha AXP architecture. The project goal to achieve best-in-class system performance for cost-focused applications was realized by integrating a high-performance CPU, a high-bandwidth memory controller, a phase-locked loop, and a low-cost package, and by being the first microprocessor in the industry that connects directly to the PCI bus. A pseudorandom verification strategy that leveraged an architectural reference with an I/O stream was a key decision that helped achieve first-pass silicon success on a compressed 10-month schedule.

Acknowledgments

The authors would like to thank the entire DECchip 21066 team. The efforts of all those involved in the microprocessor architecture, the design, the implementation, the logic verification, and the layout made it possible to bring this project to fruition.

References and Note

1. R. Sites, ed., *Alpha Architecture Reference Manual* (Burlington, MA: Digital Press, 1992).
2. *Peripheral Component Interconnect (PCI)* (Hillsboro, OR: PCI Special Interest Group, rev. 2.0, April 30, 1993).
3. D. Dobberpuhl et al., "A 200-MHz 64-bit Dual-issue CMOS Microprocessor," *Digital Technical Journal*, vol. 4, no. 4 (Special Issue 1992): 35–50.
4. B. Zetterlund, J. Farrell, and T. Fox, "Microprocessor Performance and Process Complexity in CMOS Technologies," *Digital Technical Journal*, vol. 4, no. 2 (Spring 1992): 12–24.
5. *Intel Pentium Processor Data Book*, vol. 1 (Santa Clara, CA: Intel Corporation, 1993).
6. *Intel486 DX Microprocessor Data Book* (Santa Clara, CA: Intel Corporation, 1991).
7. G. Darcy III et al., "Using Simulation to Develop and Port Software," *Digital Technical Journal*, vol. 4, no. 4 (Special Issue 1992): 181–192.

8. W. Anderson, "Logical Verification of the NVAX Chip Design," *Digital Technical Journal,* vol. 4, no. 3 (Summer 1992): 38–46.

9. D. Su et al., "Experimental Results and Modeling Techniques for Substrate Noise in Mixed-Signal Integrated Circuits," *IEEE Journal of Solid State Circuits,* vol. 28, no. 4 (April 1993): 420–430.

10. F. Gardner, *Phaselock Techniques,* 2d ed. (New York: John Wiley & Sons, 1979).

11. R. Best, *Phase-Locked Loops* (New York: McGraw-Hill, 1984).

12. P. Gray and R. Meyer, *Analysis and Design of Analog Integrated Circuits* (New York: John Wiley & Sons, 1984).

13. M. Wakayama and A. Abidi, "A 30-MHz Low-Jitter High Linearity CMOS Voltage-Controlled Oscillator," *IEEE Journal of Solid State Circuits,* vol. SC-22, no. 6 (December 1987): 1074–1081.

14. K. Kurita et al., "PLL-Based BiCMOS On-Chip Clock Generation for Very High-Speed Microprocessors," *IEEE Journal of Solid State Circuits,* vol. 26, no. 4 (April 1991): 585–589.

15. S. Miyazawa et al., "A BiCMOS PLL-Based Data Separator Circuit with High Stability and Accuracy," *IEEE Journal of Solid State Circuits,* vol. 26, no. 2 (February 1991): 116–121.

16. I. Young, J. Greason, and K. Wong, "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors," *IEEE Journal of Solid State Circuits,* vol. 27, no. 11 (November 1992): 1599–1607.

17. N. Nguyen and R. Meyer, "Start-up and Frequency Stability in High-Frequency Oscillators," *IEEE Journal of Solid State Circuits,* vol. 27, no. 5 (May 1992): 810–820.

18. J. Melsa and D. Schultz, *Linear Control Systems* (New York: McGraw-Hill, 1969).

19. SPICE is a general-purpose circuit simulator program developed by Lawrence Nagel and Ellis Cohen of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

20. W. Hayt et al., *Engineering Circuit Analysis* (New York: McGraw-Hill, 1978).

Referees, January 1993 to February 1994

The editors acknowledge and thank the referees who have participated in a peer review of the papers submitted for publication in the Digital Technical Journal. *The referees' detailed reports have helped ensure that papers published in the* Journal *offer relevant and informative discussions of computer technologies and products. The referees are computer science and engineering professionals from academia and industry, including Digital's consulting engineers.*

Alan Abrahams, Digital
Glenn Adams, Metis Technology, Inc.
Takao (Tim) Aihara, Digital
Dimitri A. Antoniadis, MIT
Guillermo Arango, Schlumberger Laboratory for Computer Science
Robert M. Ayers, Adobe Systems, Inc.
Wael Bahaa-el-Din, Digital
David Birdsall, Tandem Computers Inc.
Jorge L. Boria, Schlumberger Laboratory for Computer Science
Joseph Bosurgi, Univel
V. Michael Bove, Jr., MIT Media Lab
Mark Bramhall, Digital
Ronald Brender, Digital
Michael Brey, Digital
David Carver, MIT Laboratory for Computer Science
Chungmin (Melvin) Chen, University of Maryland
Lou Cohen
Andrew M. Daniels, Xerox Corporation
Scott H. Davis, Digital
Alexis Delis, Queensland University of Technology
Kevin P. Donnelly, Gaelic Medium School
Bob Fleischer, Digital
John Forecast, Digital
Frank Fox, Digital
Asmus Freytag, Microsoft Corporation
Ophir Frieder, George Mason University
Denis Garneau
Peter Hayden, Digital
James Horning, Digital
Hai Huang, Digital
Vania Joloboff, Open Software Foundation, Inc.
Capers Jones, Software Productivity Research, Inc.
Larry Jones, Digital
Ashok Joshi, Digital
Nancy Kronenberg, Digital
Alberto Leon-Garcia, University of Toronto
Ian Leslie, University of Cambridge
Thomas Levergood, Digital
Peter Lockemann, Karlsruhe University
David Marca, Digital
William McKeeman, Digital
Gary W. Miller, International Business Machines Corporation
James H. Morris, Carnegie-Mellon University
Andre Nasr, Digital
Stephen Neupauer, Digital
P. J. Plauger
George C. Polyzos, University of California
Edward Prentice, Digital
Roger S. Pressman, R. S. Pressman & Associates, Inc.
Ron Radice, Carnegie-Mellon University
Sridhar Raghavan, Digital
Yoav Raz, Digital
Rahul Razdan, Digital
Dave Redell, Digital
Frank Rojas, International Business Machines Corporation
Robert C. Rose, Digital
Chris Schmandt, MIT Media Lab
Peter Spiro, Digital
David Staelin, MIT
Larry Stewart, Digital
Mani Subramanyam, Ingres Corporation
Kouichi Suzuki, Nippon Telegraph and Telephone Corporation
Dave Taylor, SunWorld
David L. Tennenhouse, MIT Laboratory for Computer Science
Charles P. Thacker, Digital
David W. Thiel, Digital
Win Treese, Digital
David Wecker, Digital
Andy Wilson, Digital
John Wroclawski, MIT Laboratory for Computer Science
Wai Yim, Digital

Further Readings

Digital Research Laboratory Reports

Reports published by Digital's research laboratories can be accessed on the Internet through the World Wide Web or FTP. For access information on the electronic or hard-copy versions of the reports, see http://gatekeeper.dec.com/hypertext/info/cra-reports.html

Technical Papers by Digital Authors

The following technical papers were written by Digital authors:

R. Abugov, "From Trend Charts to Control Charts: Setup Tests for Making the Leap," *IEEE Transactions on Semiconductor Manufacturing* (May 1993).

B. Archambeault, "FDTD Modeling of the Resonant Characteristics of Realistic Enclosures," *Applied Computational Electromagnetics Society Symposium* (March 1993).

B. Archambeault and L. Lemaire, "Optimization of the FDTD Numerical Technique for EMI Modeling," *Applied Computational Electromagnetics Society Symposium* (March 1993).

B. Archambeault and R. Mellitz, "EMI Predictions in NEC Using Wire Mesh Boxes," *Applied Computational Electromagnetics Society Symposium* (March 1993).

N. Awad, "Connector Manufacturing—Process Certification," *American Society for Quality Control Forty-ninth Annual RSQC Conference* (March 1993).

S. Batra and A. Torabi, "Monte Carlo Simulations of Shielded MR Head," *IEEE International Magnetics Conference (INTERMAG '93)* (April 1993).

C. Brench, "Applications of MiniNEC to EMI Modeling," *Applied Computational Electromagnetics Society Symposium* (March 1993).

D. Butchart and E. Zimran, "Performance Engineering Throughout the Product Life Cycle," *IEEE 1993 CompEuro Proceedings: Computers in Design, Manufacturing, and Production* (May 1993).

K. Curewitz, "Practical Prefetching via Data Compression," *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '93)* (May 1993).

N. Doraswamy, "Chitra: Visual Analysis of Parallel and Distributed Programs in the Time, Event, and Frequency Domains," *IEEE Transactions on Parallel and Distributed Systems* (November 1992).

B. Doyle, J. Faricelli, and K. Mistry, "Characterization of Oxide Trap and Interface Trap Creation During Hot-Carrier Stressing of n-MOS Transistors Using the Floating-Gate Technique," *IEEE Electron Device Letters* (February 1993).

B. Doyle, K. Mistry, and D. Krakauer, "Examination of Oxide Damage During High-Current Stress of n-MOS Transistors," *IEEE Transactions on Electron Devices* (May 1993).

B. Doyle and A. Philipossian, "p-Channel Hot-Carrier Optimization of RNO Gate Dielectrics Through the Reoxidation Step," *IEEE Electron Device Letters* (April 1993).

R. Dutta, "Algorithm for Wire Sizing of Power and Ground Networks in VLSI Designs," *Journal of Circuits, Systems, and Computers* (June 1992).

M. Elbert, G. Kushner, C. Mpagazehe, and T. Weyant, "Stress Testing—Our Approach," *IEEE Thirty-first Annual Spring Reliability Symposium* (April 1993).

C. Gordon, "Time-Domain Simulation of Multiconductor Transmission Lines with Frequency-Dependent Losses," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (November 1992).

C. Huang, N. Arora, and J. Faricelli, "A New Technique for Measuring MOSFET Inversion Layer Mobility," *IEEE Transactions on Electron Devices* (June 1993).

J. Kohl, "HighLight: Using a Log-structured File System for Tertiary Storage Management," *Proceedings of the Winter 1993 USENIX Conference* (January 1993).

Y. Kra, "A Cross-Debugging Method for Hardware/Software Co-Design Environments," *Thirtieth Design Automation Conference* (June 1993).

K. MacNeal, "Analytical Tools for Adhesive Process Development," *Society of Plastic Engineers Conference Proceedings (ANTEC '93)* (May 1993).

D. McLeish, V. Mastro, and S. Artsy, "Requirements for a Repository Information Model," *Database Newsletter* (March/April 1992).

E. McLellan, "The Alpha AXP Architecture and 21064 Processor," *IEEE MICRO* (June 1993).

B. Mirman, "Interlaminar Stresses in Layered Beams," *ASME Journal of Electronic Packaging* (December 1992).

K. Mistry and B. Doyle, "The Characterization of Hot-Carrier Damage in p-Channel Transistors," *IEEE Transactions on Electron Devices* (December 1992).

J. Palczynski, "Current Diverter—A Novel Device to Regulate Multiple Outputs," *IEEE Eighth Annual Applied Power Electronics Conference (APEC '93)* (March 1993).

K. Ramakrishnan, "Performance Issues in Designing Network Interfaces: A Case Study," *Fourth IFIP Conference on High Performance Networking* (December 1992).

K. Ramakrishnan, "Scheduling Issues for Interfacing to High Speed Networks," *IEEE Global Telecommunications Conference* (December 1992).

Y. Raz, "Commitment Ordering Based Distributed Concurrency Control for Bridging Single and Multi Version Resources," *IEEE Third International Workshop in Data Engineering: Interoperability in Multidatabase Systems (RIDE-IMS '93)* (April 1993).

R. Razdan, "HCNC: High Capacity Netlist Compare," *IEEE Custom Integrated Circuits Conference* (May 1993).

S. Rege and R. Kalkunte, "FDDI Adapter Design," *FDDI: Technology and Applications* (New York, NY: John Wiley and Sons, 1992).

S. Sathaye, "Logical Analysis and Control of Time Petri Nets," *Thirty-first IEEE Conference on Decision and Control* (December 1992).

S. Sathaye, "Scheduling Real-Time Communication on Dual-Link Networks," *IEEE Real-Time Systems Symposium* (December 1992).

J. Sauber, "A Low-Cost, High I/O Separable Module Interconnect Design," *IICIT Twenty-fifth Annual Connector and Interconnection Technology Symposium (CONN-CEPT '92)* (September 1992).

J. Sauber and J. Seyyedi, "Predicting Thermal Fatigue Lifetimes for SMT Solder Joints," *ASME Journal of Electronic Packaging* (December 1992).

J. Seyyedi, "Thermal Fatigue Behaviour of Low Melting Point Solder Joints," *Soldering & Surface Mount Technology* (Journal of the SMART Group) (February 1993).

A. Shvartsman, "An Efficient Write-All Algorithm for Fail-Stop PRAM Without Initialized Memory," *Information Processing Letters* (December 1992).

A. Shvartsman, "An Historical Object Base in an Enterprise Management Director," *IEEE/IIP Third International Symposium on Integrated Network Management* (April 1993).

R. Sites, "Alpha AXP Architecture," *Communications of the ACM* (February 1993).

H. Soleimani, "Modeling of High-Dose Ion Implantation-Induced Dopant Transient Diffusion, and Dopant Transient Activation in Silicon (Boron and Arsenic Diffusion)," *Journal of the Electrochemical Society* (November 1992).

K. Somalwar, "Scheduling Parallel I/O Operations in Multiple Bus Systems," *Journal of Parallel and Distributed Computing* (December 1992).

S. Susswein, "Parallel Path Consistency," *International Journal of Parallel Programming* (December 1991).

J. Thottuvelil, F. Tsai, and D. Moore, "Application of Switched-Circuit Simulators in Power Electronics Design," *IEEE Eighth Annual Applied Power Electronics Conference (APEC '93)* (March 1993).

M. Tsuk, "A Nonlinear Transmission Line Model for Superconducting Stripline Resonators," *IEEE Transactions on Applied Superconductivity* (March 1993).

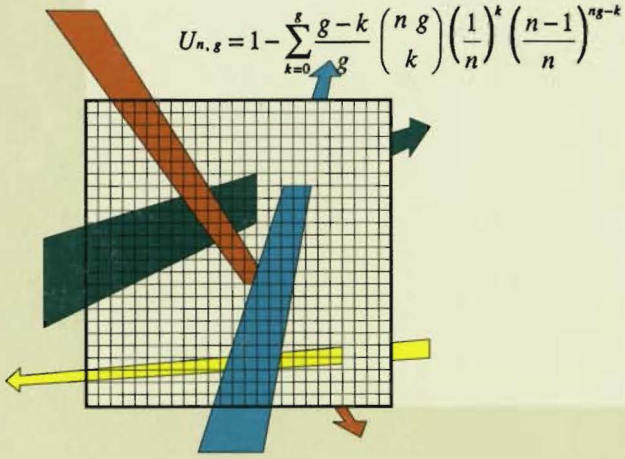
R. Ulichney, "Bresenham-Style Scaling," *IS&T's (The Society for Imaging Science and Technology) Forty-sixth Annual Conference* (May 1993).

A. Vitale, "Hardware and Software Aspects of a Speech Synthesizer Developed for Persons with Disabilities," *Journal of the American Voice I/O Society* (March 1993).

J. Yang, "Reliability Evaluation of a High-Density Thin-Film Multichip Module," *IEEE Reliability and Maintainability Symposium* (January 1993).

K. Zinke, R. Abugov, and W. Lauer, "A Die Level Strategy to Quantify Device Yield as a Function of Surface Particles," *Institute of Environmental Sciences 1993 Proceedings* (May 1993).

digital™



ISSN 0898-901X