Performance Analysis of a High-speed FDDI Adapter

By Ramsesh S. Kalkunte

Abstract

The DEC FDDIcontroller 400 host-to-FDDI network adapter implements real-time processing functionality in hardware, unlike conventional microprocessor-based designs. To develop this high-performance product with the available technological resources and at minimal cost, we optimized the adapter design by creating a simulation model. This model, apart from predicting performance, enabled engineers to analyze the functional correctness and the performance impact of potential designs. As a result, our implementation delivers close to ultimate performance for an FDDI adapter and surpasses the initial project expectations.

As high-performance systems become available and the use of distributed computing proliferates, the need for high-performance networks increases. Faster adopting fiber distributed data interface (FDDI) local area network (LAN) technology as a follow-on to Ethernet, Digital recognized the need to build an industry-leading network adapter to service its high-performance platforms. As a result, we designed and developed the DEC FDDIcontroller 400 product. To track the adapter performance through the design and development stages, we created a simulation model; our objective was to ensure that the device met our performance goals. This paper begins with a description of the DEC FDDIcontroller 400, followed by a brief historical perspective and statement of the performance objectives of the adapter project. We then discuss in detail the modeling methodology and the results achieved. In addition, we present validation of these results in the form of measurements taken on prototype hardware.

interconnects are required to achieve such performance goals. Consequently, network adapters must be able to function at higher speeds. In

Performance Analysis of a High-speed FDDI Adapter

The DEC FDDI controller 400

The DEC FDDIcontroller 400, also known as the DEMFA, is a high-speed FDDI network adapter. Attached to a host machine running under either the VMS or the ULTRIX operating system, the DEMFA enables the host to communicate with other network entities through the FDDI ring. The DEMFA adapter implements Digital's proprietary XMI bus protocol and can be used with any system that has an XMI backplane.[1] Laboratory measured performance data presented later in the paper shows that the adapter hardware can sustain a practically infinite stream of frames at the full FDDI data bandwidth of 100 megabits per second (Mb/s) for frame sizes 69 bytes or larger on the receive stream and 51 bytes or larger on the transmit stream. Even the smallest, i.e., 20-byte dataless, FDDI frames can be received at 36 Mb/s and transmitted at 47 Mb/s.

The DEMFA is an FDDI Class-B single attachment station (SAS) that interfaces to the FDDI token ring network through the DECconcentrator 500. A port driver resident in the host controls the DEMFA port. The port, the port driver, and the adapter hardware implement DECnet, the transmission control protocol with the internet protocol (TCP/IP), and local area transport (LAT).[2] Figure 1 shows a typical network configuration using the DEC FDDIcontroller 400 adapter with other Digital FDDI products.

The XMI bus is capable of transferring data at rates up to 800 Mb/s and can serve as either a CPU-to-memory interconnect, e.g., in the VAX 6000 platform, or an I/O bus, e.g., in the VAX 9000 platform.[3,4] Also, Digital plans to include the XMI bus in future systems.

FDDI is a timed-token, fiber-optic ring that provides a network data bandwidth of 100 Mb/s.[5] In addition to this high data rate, the advantages of low signal attenuation, low noise susceptibility, high security, and low cost (as the technology matures) will make FDDI a popular interconnect of the 1990s.[6]

the American National Standards Institute (ANSI) data link and physical layer functionality for FDDI LANs. This foundation supports user protocols such as the Open Systems Interconnection (OSI),

Historical Perspective and Performance Objectives of the DEMFA

With the advent of high-performance systems and distributed computing strategies, the need for high-performance networking options has increased. Traditionally, I/O adapters have been built to serve the current performance needs. As a consequence, such adapters offer little or no network performance scalability to accommodate future increases in demand. Scalability is important to ensure that the adapter does not become a bottleneck when such demands exist. Nonscalable adapters become obsolete, and the resulting frequent hardware upgrades increase system cost.

The first Ethernet adapters, which complied with the IEEE 802.3 standard, were built in the early 1980s. Only recently do adapters exist that can process frames at the maximum Ethernet throughput rate of 10 Mb/s.[7] As mentioned earlier, FDDI has the capability of supporting speeds an order of magnitude higher than Ethernet. Since the header in an FDDI frame is three times smaller than that for Ethernet, bytes and larger at 100 Mb/s, i.e., the adapter would be able to process approximately 80,000 frames per second (frames/s). Also, twenty microseconds was deemed an acceptable adapter latency for the smallest FDDI frames. Considering the relatively small number of frames a host system can process today, these adapter criteria represented an ambitious goal-one which would make a product with high-performance scalability as faster CPUs became available.

Performance Modeling Considerations

During the development of a high-performance product, changes in architectural functionality, technology constraints, and cost considerations can result in design modifications. It is desirable to track the performance of the product through its development to understand the impact of such modifications.

The DEMFA consists of many hardware entities that perform the desired adapter functions.[8] Although such hardware adapters have the obvious advantage of superior performance over conventional, i.e., microprocessor-

FDDI frame arrival rates can be as much as 30 times the Ethernet arrival rate. Considering the various constraints, Digital set out with the goal to build an FDDI adapter that could process frames 150

based adapter cards, this advantage does not come without the risks associated with hardwired logic. Such risks have a negative impact on project budget and schedule and necessitate a risk

management strategy to ensure that product goals are successfully met. Performance modeling of the adapter and extending the use of such modeling to evaluate various designs formed part of this strategy. The following subsections describe the goals and tasks of the DEMFA performance modeling.

Goals

The set of performance modeling goals for the DEMFA evolved throughout the development process. Three major goals were performance projection, buffer sufficiency analysis, and design testing through simulation.

Performance Projection. In the early phases of the design, the primary goal of the model was to project the adapter performance. This prediction gave us confidence that the design could meet our performance expectations.

Buffer Sufficiency Analysis. Buffer capacity plays an important part in the performance of a design. Whereas too much of this resource is wasteful, too little has a negative effect on performance. It was critical to determine the extent of buffering necessary to attain the desired target performance at the least cost. The performance model considered the dependencies on this resource. The amount of buffering was varied and the effects of such variation, manifested in the simulation results, were analyzed. Using these results as input to a cost/benefits equation helped the designers make

The performance model served as a platform that could be enhanced to solve these more complex problems by simulation. Designs were analyzed to determine their impact on adapter performance. Because the simulation methodology afforded greater testability, we were able to make the designs more robust and to answer design questions in a significantly shorter time than other methods. Consequently, modifications to the hardware were made at an early design stage and at negligible cost.

Tasks

To accomplish performance modeling, we faced the following basic tasks: choosing the metrics, defining the workload, and deciding on a modeling methodology. Relevant metrics to measure the performance of a product are crucial. We chose metrics that are simple to understand and provide insight into the behavior of the product. Also, areas in which workload development is required must be identified and investigated in detail. An incorrect workload invalidates all performance data. And the methodology used to model the system must be well thought-

intelligent decisions concerning buffer capacity.

Design Testing through Simulation. As development progressed, important design issues arose that could not be solved by simple analysis.

out beforehand, so that the model is accurate and also flexible enough to be easily changed.

Definition of Metrics. The main performance metrics used were throughput and frame latency. Throughput is the rate at which frames are processed and is measured in megabits per second or frames per second. The units can be converted easily from one to the other, if the average frame size is specified. In this paper, throughput is expressed in megabits per second.

Frame latency is the elapsed time measured in microseconds between the time at which a frame is queued for service at a facility and the time at which the service is completed. The following descriptions illustrate the approach used to measure receive and transmit latency. The host receives frames from and transmits frames to the FDDI ring. Receive frame latency is the time elapsed between (1) the arrival of the last bit of the frame into the adapter from the FDDI ring and (2) the time the frame becomes available to the host for processing. Transmit frame latency is the elapsed time between (1) the time the adapter starts processing a frame from the host and (2) the exit time of the first bit of the frame from the adapter destined for the insight into the adapter behavior. For the context of this paper, we consider the DEMFA processing pure frame streams only, i.e., the expressions "receive throughput" and "receive latency" refer to a pure receive stream of frames containing no transmit frames. Similarly, "transmit throughput" and "transmit latency" refer to a pure transmit stream of frames.

Workload Definition. Using a relevant traffic workload is very important in any simulation model. Since most systems are workload-sensitive, defining an incorrect workload may result in irrelevant data. We identified two areas in which we needed to define workloads. We then characterized the traffic patterns and built a workload model for performance simulation based on these patterns.

- Frame receive and transmit workloads. The receive and transmit workloads are stimuli for the performance simulation. These workloads mimic traffic due to frame arrival on the FDDI ring (i.e., the receive workload) or frame transmission from the host (i.e., the transmit workload). The receive workload

FDDI ring.

The adapter can process transmit and receive frames simultaneously. We defined performance metrics to analyze a variety of traffic scenarios to gain

model generates frames which the DEMFA model receives, whereas the transmit workload acts as a source of frames to be transmitted by the DEMFA model on

the FDDI ring. These workloads must be characteristic of actual FDDI traffic. Since FDDI LANs did not exist when the DEMFA was in the development stage, we used our experiences with Ethernet to derive these workloads, as we explain in greater detail in the FDDI Token Ring section.

- XMI traffic workload. Apart from the DEMFA traffic, there may be other traffic on the XMI bus due to CPU-to-memory transactions

or from other I/O adapters attached to the system. The load on the XMI bus impacts the performance of the DEMFA. Consequently, we designed a workload model to mimic the traffic pattern on the bus. We based our model on the traffic patterns observed for real XMI bus traffic. The performance of DEMFA may degrade as this traffic increases because the DEMFA traffic and the non-DEMFA traffic consume common resources. The other traffic is referred to as the XMI interference workload. The XMI Workload Generator section describes the

allows changes to be made easily. The SIMULA language implements the simulation model.[9] The simulation-class and queuing constructs in this language are tailored to help simulation and modeling.[10,11] The object-oriented structures present other advantages to model development. A debug procedure coded into the model prints status information about all the queues in the model. This information helped us trace the path of frames through the system.

One important first step in designing a simulation model is to determine the detail at which to model. Two factors that influence the level of detail are the

- Existing knowledge of the design. Usually, information gathered from the behavioral and analytical models of a design helps to make a performance model abstraction. Designs with behavior that cannot be analyzed by these lower-level models have to be modeled in greater detail.

- Expectation of performance model accuracy. Typically, a performance model predicts results

model for this workload.

Modeling Methodology. The simulation model has a hierarchical design to allow the construction of smaller, more manageable blocks, i.e., submodels. The structure also

accurate to within ±10.0 percent of the performance that would be achieved with the actual hardware.

During the design phase, behavioral and structural models of hardware were in development. This hardware was partitioned across important functional boundaries. Hardware within these boundaries would be modeled and tested thoroughly by the respective development engineers. Hence, to include details of these pieces of hardware in our model would have resulted in redundant effort. Since the interfaces and the gross functionality of the hardware within these boundaries are relevant to performance, we did include these components in our model. Existing hardware components, such as the FDDI chip set, were grouped together before being modeled for functionality. Each submodel was designed and tested separately to ensure conformity to the functionality and performance of other behavioral and structural models. This strategy resulted in the base-level performance model that we used to generate preliminary performance data for the DEMFA.

As development progressed, we encountered design changes of various a part and generalized to the adapter system environment in which the piece operates. Models that represent the changes were included and interfaced as submodels. These submodels served the dual purposes of testing the new design and of improving the accuracy of the performance model.

Design of the Simulation Model

The performance simulation model consisted of the following major components:

- FDDI ring
- FDDI chip set and parser
- Packet memory controller
- Host interface
- XMI system
- Host system

The base-level model evolved over time, as we gained insight into the behavior of the individual components and defined workloads. The model evolved further to support the need to analyze new designs through simulation. This section briefly describes the components of the final model, as listed above.

FDDI Token Ring

The FDDI token ring was modeled to act as a source

complexities. Simple design changes resulted in very small changes in the performance model. But larger and more complex design changes required that we investigate behavior both specific to the piece of hardware of which the design is

of received frames and as a sink of transmit frames. Gross functionality for the remainder of the FDDI nodes and network components was desirable. Consequently, we designed a black-box model for the FDDI ring that provides two-way interaction with the FDDI

chip set and parser model. This FDDI model allocates time on the FDDI ring for transmit and receive transactions. The model also controls a receive workload generator when frames are received by the adapter.

The receive workload generator is an analytical model used to create different patterns of receive traffic to the DEMFA. The parameters input to this workload model are the average frame size, the frame-size distribution, the frame type, the load, and the number of back-to-back frame arrivals (i.e., the burst rate or "burstiness" of the frame arrivals). We varied these parameters to generate desired workloads.

The average frame size and frame-size distribution parameters generate different size frames.

Actual frame sizes can be specified as normally or exponentially distributed about the mean or as constant. The workload model can generate station management (SMT), LLC SNAP /SAP, or LLC non-SNAP/SAP frame types and can create a load between 0 and 100 Mb/s. If workloads are less than the peak FDDI bandwidth, i.e., 100 Mb/s, the traffic as seen in realistic networks.

Several studies had been conducted on large Ethernet LANs within Digital; a case study by D. Chiu and R. Sudama is one example.[12] We analyzed the results from these studies to understand the frame-size distribution in such networks. From the analysis we concluded that

- Frame sizes on the networks are related to user protocols. Frames in a test sample were distributed about a few discrete frame sizes (i.e., modes of the distribution) rather than over a wide range of frame sizes.
- The probability function of the frame sizes near each mode can be approximated as a normal distribution centered about the mode.

A composition analysis of the measurements provided different modal mean sizes, standard deviations, and the probabilities of frames belonging to the different modes. We used these values to statistically create Ethernet network traffic. For our performance measurements, it was necessary for us to change this traffic pattern

the frame arrival pattern can be specified as an exponential, constant, or normal distribution. The model can generate a wide range of synthetic traffic patterns, but to obtain credible performance results, we characterized appropriately to reflect the differences that exist between FDDI LANs and Ethernet LANs. The FDDI frame header is smaller than the Ethernet header, and the largest FDDI frame is approximately three times the size of the

largest Ethernet frame. We factored these changes into the Ethernet model to produce an FDDI workload model. The FDDI workload has either four or five modes.

The four-mode distribution contained a majority of frames grouped around 60, 576, 1518, and 4496 bytes. The standard deviations of the frames around these mean values were 22, 5, 2, and 2 bytes, respectively. The frame volumes at these modal values represented contributions of 29 percent, 67 percent, 3 percent, and 1 percent, respectively, to the total load.

The five-mode frame sizes were grouped around 33, 80, 576, 1518, and 4496 bytes. The standard deviations of the frames around these means were 1, 20, 5, 2, and 2, respectively. The frame volumes at these modes contributed 26 percent, 15 percent, 55 percent, 3 percent, and 1 percent, respectively, to the total load.

In the above FDDI workload model, the mode of 1518 bytes is determined by the Ethernet network's maximum frame-size capacity and, similarly, the mode of 4496 bytes is determined by the FDDI network's maximum frame-size capacity. These network. We considered different contributions and found their effect on adapter throughput to be negligible. Therefore, only one case for each workload is presented in this paper.

FDDI Chip Set and Parser

The FDDI chip set, also referred to as the FDDI corner, is the base-level technology that was part of Digital's strategy to build high-performance, low-cost data links for FDDI LANs. This chip set performs serial-to-parallel data conversion, acts as an interface to the packet memory in the data link layer, and can support a data rate of 100 Mb

/s.[13] The entire chip set, except for the ring memory controller (RMC), was modeled as a black box with a specified per-frame latency. The RMC and the associated first in, first out (FIFO) buffers for the receive and transmit stream staging were modeled in greater detail. The detail was necessary to capture any overflow or underflow conditions that might occur in the FIFO buffers. We also modeled the interaction between the transmit and receive streams. The RMC model, which served as the front end of the chip set model, was also capable

two modal frame sizes represent traffic generated by large data transfer operations, e.g., file transfers. Contributions due to these two modes vary from network to

of generating control and data transactions to perform read/write memory operations.

The parser hardware off-loads some host frame processing to the adapter. The parser reads information about a receive frame from the RMC bus and creates a forwarding vector, which is appended to the frame. This forwarding vector is used by different entities in the adapter and the host to efficiently process a frame. The parser latency to generate this vector varies with the frame type and size. The parser model helped to analyze the impact of this latency on performance. This model mimics the hardware to produce a forwarding vector for a given frame with a pertinent latency.

Packet Memory Controller

The packet memory controller (PMC) is the heart of the adapter system. The ring entry mover stage, the packet buffer memory, and the packet memory interface constitute the functionality in the PMC.[8] The PMC controls the arbitration and servicing of requests to and from memory to effect the efficient transfer of information. The PMC also controls the movement of pointers corresponding to every frame. These pointers and the associated protocol

The high throughput capability of FDDI rings can result in traffic patterns that cause a strain on the packet memory. The PMC model allowed us to study such scenarios. It is also important to analyze the working and performance of the ring entry mover, which moves frames between different interfaces by manipulating the control information of a stored frame. The control information and frame data reside in the packet memory.

Host Interface

The host interface, also called the host protocol decoder, moves data between the adapter and the host system through an XMI bus and also interfaces with the PMC. We modeled the interface to include details of the dual direct memory access (DMA) design (one channel for the receive stream and one for the transmit stream), the staging buffers associated with each DMA channel, the XMI interface, and the PMC interface. The host interface also has the capability of scheduling write operations while waiting for the delivery of read information. Priority schemes to complete such transactions, i.e.,

generate work for the RMC, the host interface, or the adapter manager.

handshake mechanisms, are important from a performance perspective and, hence, were included in the model.

XMI System

The XMI system interacts with the host system and was modeled to include details of the XMI bus and memory. This model consists of an XMI bus submodel that interfaces to the XMI end of the host interface model of the adapter. The submodel also interacts with a memory model and an XMI workload generator model. The bus submodel implements the XMI protocol.

Memory Model. The memory model was designed to generate responses to transactions that request memory. Latency for these requests is the memory access time, which includes a queue wait time. There are basically two types of systems that support the DEMFA, as shown in Figure 2. The type is determined by whether the XMI is used as the CPU bus, denoted in this paper as the XMI (CPU) bus configuration, or as the I/O bus, denoted as the XMI (I/O) bus configuration. The only difference between the two systems is memory access time. This time is greater if XMI is used as the I/O bus; there is an added latency on the read transactions performed to fetch memory from locations that are not local to the XMI bus. The memory space that is times in such systems. The model presented in this paper depicts the VAX 9000 I/O architecture and current implementation. Performance may vary with other implementations.

XMI Workload Generator. We designed the XMI workload generator to represent the load on the XMI bus, excluding traffic from the DEMFA. This load tends to have a deteriorating effect on DEMFA performance and thus, is referred to as the XMI interference workload. It was important not only to model the amount of load but also to capture the arrival pattern of this traffic. The workload model generated traffic based on three inputs: the total XMI bandwidth used by other XMI nodes, the average length of each XMI transaction, and the burst rate of the frame arrivals. Transaction lengths on XMI vary from one to five XMI cycles (i.e., 64-nanosecond cycles). The maximum number of nodes that can exist on an XMI bus is 14. Thus, the burst rate can vary from 1 to 13.

Typically, traffic on an XMI bus consists of many back-to-back transactions of various sizes. We decided to use the worst case values for both the burst rate and the transaction length in the

local to the CPU bus is accessed through another I/O adapter mechanism. Such I/O adapters, CPU buses, and main memory bandwidth all play a role in determining the access

XMI interference workload presented in this paper. The worst case burst rate is 13, and the worst case transaction length is 5 XMI cycles.

Host System

The host system consists of the CPU, disks, layered software, the operating system, the device driver, and a host workload generator. The host system was modeled in accordance with assumptions presented in the section Results from Performance Simulation. The CPU, disks, host software, and the operating system were modeled in such a way that they do not become bottlenecks during frame reception or transmission. A model of the device driver handles frame transmission and reception. The driver interacts with a host workload generator, which creates different traffic patterns for transmission. This workload generator has the same capabilities as the receive workload generator discussed in an earlier section.

Results from Performance Simulation

The data presented in this section was generated using the simulation model of the adapter.

This data represents the hardware performance of the DEMFA; system performance with the DEMFA as a component is not within

of relevant events and quantities and print out this information at the end of a simulation. As discussed previously, the hardware performance of the DEMFA varies depending upon whether the system is implemented to use the XMI bus as a CPU bus or as an I/O bus. This section presents simulation results for both uses, where appropriate.

Assumptions

For our simulation purposes, we made several assumptions. These assumptions make the results more general and bring out the hardware performance characteristics of the DEMFA, indicating the upper bounds of performance that the adapter can achieve.

CPU and Software Capabilities. The device driver and the host software do not become

bottlenecks during frame reception and transmission. We assumed that the host CPU had enough computing ability to process frames without posing as a performance bottleneck.

Memory Bandwidth. Frames sent from or received by the host result in XMI bus transactions that are written to or read

the scope of this paper. We input parameters to the simulation model that defined traffic patterns and ran simulations for a sufficient length of time to ensure that we captured steady-state behavior. The models maintain statistics from the host memory. Throughput varies with the memory implementation and interleaving. We assumed that the memory implementation and interleaving were selected such that no overloading of the memory occurs, thus

eliminating wasted bus
cycles.

Buffer Alignment and Segmentation. We assumed that data for transmission and buffers for reception were hexaword (i.e., 32-byte) aligned and that frames were unsegmented.

Simulation Traffic. No error frames or error transactions were simulated, since we assumed these to be negligible. No adapter manager traffic was simulated during the performance measurements, since these represent a very negligible fraction of the frames received during steady-state ring operation.

Throughput Measurements

Measurements were made to determine the throughput that the adapter can sustain for received and transmitted frames. It is important to understand how throughput is related to the load, the burstiness of frame arrivals, the percent XMI interference, and the frame size. This section presents the results of the throughput measurements as functions of these parameters.

Received Throughput as a Function of the Load. The graph shown in Figure 3 is the result of several experiments conducted by varying the load for 33-

an exponential arrival pattern, the throughput increases at a rate proportional to the load up to a certain point, and then gradually decreases until the load is 100 Mb/s. The decrease in throughput is caused by the loss of resources due to excessive loading.

We simulated traffic with a constant arrival pattern and conducted the same experiments. These results are also shown in Figure 3. Observe that the point of maximum throughput and the rate at which the throughput decreases after reaching the maximum vary with the arrival pattern of traffic. After performing experiments on other frame sizes, we concluded that there is no fixed relationship between the maximum achievable throughput and the throughput at FDDI saturation (i.e., 100-Mb /s load). Also, there is graceful degradation in throughput after the peak.

Receive Throughput for Four- and Five-mode Workloads. We measured adapter receive throughput for four- and five-mode workloads with a load of 100 Mb/s. The XMI interference workload was varied, and the results

byte received frames. The frame arrival rates depend on the load and the arrival rate distribution. As mentioned earlier, the model is capable of simulating traffic with different arrival patterns. Figure 3 shows that, with the workload at 100 Mb/s, are presented in Figure 4. The adapter can receive if the XMI interference workload remains moderate. Figure 4 also shows that there is very little difference in performance between the four- and

five-mode workloads. Large frames constitute a major part of both workloads, and larger frames can be easily supported by DEMFA at full FDDI data bandwidth.

Receive Throughput as a Function of Frame Size. Figure 5 shows the throughput as a function of the frame size and the XMI interference workload, with DEMFA attached to an XMI (CPU) bus. Smaller frames have a lower throughput rate than larger ones because of high control /data overhead. Since control transactions consume bandwidth, the bandwidth available for data movement is reduced. Consequently, the overall throughput rate is lower. Another reason for lower adapter throughput is the XMI utilization by traffic from other nodes on the XMI bus. This XMI interference results in less available XMI bandwidth for the adapter and hence, less throughput.

The adapter throughput for an XMI (I/O) bus configuration differs only slightly from that for an XMI (CPU) bus configuration. Any differences that exist are for frames smaller than 64 bytes, since the adapter experiences a per-frame latency cost because the

Transmit Throughput for Four- and Five-mode Workloads. Figure 6 illustrates the transmit throughput for a four-mode workload as a function of the XMI interference. We performed simulations to obtain throughput data for the DEMFA when attached to an XMI (CPU) bus or to an XMI (I/O) bus. Throughput for the XMI (CPU) bus configuration is 100 Mb/s and is insensitive to low, XMI interference loads. Whereas, XMI (I/O) bus configuration measurements are negatively affected by all levels of XMI interference traffic. The higher read latency that is inherent to an XMI (I/O) bus configuration degrades further with increasing interference traffic. In addition the degradation appears to be linear. The throughputs observed for the five-mode workloads are very similar to the data shown in Figure 6.

Transmit Throughput as a Function of the Frame Size. Figure 7 shows the throughput as a function of the frame size when the DEMFA is attached to an XMI (CPU) bus. Throughput is also presented for various XMI interference workloads. As in the case of receive throughput, transmit throughput degrades as the

memory is not local to the XMI bus.	frame size decreases and the XMI interference load increases. This degradation is again attributed to high control/data overhead and lower XMI bandwidth availability.

Latency Measurements

Latency, as it relates to the DEMFA, is explained in the Definition of Metrics section. We measured the latency for receive and transmit frames. Frame latency consists of two components: the active component, which contributes to the time when the frame or a portion thereof is being processed at a service center (also called the service time); and the passive component, which is the time when the frame or a portion thereof waits for access to the service center. All latency data presented in this section represents averages across a large number of samples. When measuring the latency of a frame, we applied the maximum load that can be sustained continuously for that frame size and type.

Receive Latency as a Function of the Frame Size. Figure 9 represents the receive latency data as a function of the frame size for an XMI (CPU) bus configuration. Latency is also presented for various XMI interference levels. We present performance data for only one XMI configuration because there is little variation between the results for linearly with increased XMI interference.

Transmit Latency as a Function of the Frame Size. Figure 10 presents transmit latency results for an XMI (CPU) bus configuration and Figure 11 presents the results for an XMI (I/O) bus configuration. The latency was measured as a function of the frame size for various XMI interference workloads. Transmit latency is more sensitive to the system type and to the XMI interference workload because most XMI transactions that constitute transmit traffic are read operations. There is a distinctly higher latency cost associated with these transactions in the XMI (I/O) bus configuration as compared to the XMI (CPU) bus configuration. As in the case of receive latency, the transmit latency degrades with XMI interference.

the XMI (CPU) bus and XMI (I/O) bus configurations. Both frame size and latency are plotted using logarithmic scales. The data illustrates that XMI latency increases

Performance Measurements with the Prototype DEMFA

The intent of performing measurements with the prototype DEMFA was twofold. First, we wanted to confirm the performance predictions arrived at through simulation. And second, we wanted to measure some features that we did not implement in the model, either because they were not quantifiable or because they were too complex to model. Again, we present only hardware performance measurements; system performance with the DEMFA is beyond the scope of this paper.

Measurement Setups

The experimental configuration required to perform the measurements on the prototype DEMFA is shown in Figure 12. This configuration consists of a VAX 6000 processor connected to a DECconcentrator 500. The VAX 6000 system has an XMI backplane. The DEMFA occupies one of the slots in the XMI backplane and is part of the XMI (CPU) bus configuration in this system.

An FDDI tester is also attached to the DECconcentrator 500 and acts as a source of frames. The FDDI tester is a useful tool for testing the DEMFA product; the tester is capable of transmitting traffic at 100 Mb/s and can generate frames of various sizes and types with different destination addresses. A standalone software driver and operating system runs on the VAX 6000 system and is used for DEMFA hardware performance tests. A logic analyzer is used to measure elapsed time and count events.

Throughput Measurements The device driver measures receive and transmit throughput and is designed to perform minimal processing for each frame.

Receive Throughput Measurements. We measured the receive throughput by sending a continuous stream of frames at 100 Mb/s from the FDDI tester to the DEMFA. We varied the frame size for the tests and ran each test for a length of time sufficient to verify data convergence.

We compared the prototype measurements with the modeled results for receive throughput as a function of the frame size for an XMI (CPU) bus configuration. This validation of the receive throughput results is shown in Figure 13. The hardware measurements demonstrate that the adapter can receive frame sizes above 69 bytes at 100 Mb/s. Throughput degrades for smaller frame sizes. These measurements closely validate the modeled results. The throughput for the performance model demonstrates that the DEMFA can continuously receive frames greater than 65 bytes at 100 Mb/s. There is a slight difference between the measured and modeled results at the lower frame throughput and is therefore acceptable.

Transmit Throughput Measurements. To measure the transmit throughput, we forwarded frames from the driver to the FDDI ring at the maximum possible rate. The throughput was calculated from the number of frames that could be sent in a unit of time. The adapter can transmit frames larger than 51 bytes at 100 Mb/s. Transmit throughputs measured in the laboratory validate the modeled results as closely as the receive throughput validation results shown in Figure 13. The modeled throughput results were lower than the measured results because we used a conservative approach to modeling the memory latency.

Multisegmented and Misaligned Frames. Segmentation and alignment of transmit frame buffers in host memory is variable. Typically, frames consist of two segments, the first containing the frame header information and the second containing the data. Since the DEMFA must access control and data separately, segmentation makes this process less efficient, from a hardware perspective, than if the data and control

sizes because residual XMI interference traffic exists in the measured system. This experimental error is unavoidable, but the difference is a small percentage of the total

information exist in the same buffer. Also, buffers may be aligned to start on different byte boundaries. Since the DEMFA transactions begin on hexaword (i.e., 32-byte) boundaries,

hexaword alignment of frame data in the host buffers is the most efficient arrangement from the adapter's perspective. We measured throughput with unsegmented and two-segmented frames, and with frames aligned on longword, quadword, and hexaword byte boundaries. Segmentation and alignment variations cause negligible throughput degradation for frames 64 bytes or larger.

Latency Measurements

We used the logic analyzer to measure the frame latency. The logic analyzer responds to signals that indicate the starting and ending times for processing a frame. The difference between these two times is the frame latency. The events were chosen such that the measurements conformed to the definition of latency as described in the Definition of Metrics section.

Note that the traffic pattern used to measure latency in this section differs from the workload illustrated in the section Performance Results from Simulation. Here, a single frame was received or transmitted, and we measured latency due to that frame only. Whereas previously, we used the because of the practical difficulty to perform latency measurements on a large number of frames.

simulation model to measure latency as an average across a large number of frames representing a load equal to the maximum sustainable adapter throughput. The workloads differ

Receive Latency. The receive frame latency predictions from the performance model and adapter service time measurements taken from the prototype hardware are shown in Figure 14. These latency measurements validate the model predictions in a way similar to that for the throughput measurements.

Transmit Latency. We also compared transmit latency measurements to predictions from the performance model and found these measurements to approximate the modeled results. But actual latency measurements were slightly lower than the modeled results, again due to a conservative modeled latency.

Conclusions

The performance model was intended to track the performance of the prototype hardware to an accuracy of ±10.0 percent. The comparisons between modeled and measured results demonstrate that the model actually surpasses our goal. The measured performance for the XMI (I/O) bus configuration using a VAX 9000 system validated the modeled results and pessimistic memory latency assumptions for transmit frames.

Throughput due to the four- and five-mode workloads is nearly the same. The average frame size for these distributions is 496 bytes and 487 bytes, respectively. Thus, throughput is a function of the frame size and independent of the number of modes that exist in the workload. Also, this data leads to the conclusion that the DEMFA may never pose as a performance bottleneck in a real network environment.

For the simulation, we chose an XMI workload with an extremely high burst rate. Actual XMI systems may result in better throughput than that presented in this paper. The resources required to create XMI workload variations are not easily accessible, so we did not perform measurements on the prototype adapter under different workload conditions. But since other measurements validated the model predictions so closely, measuring performance with varied XMI workloads proved unnecessary.

Validation of the results

as closely as did the corresponding results for the XMI (CPU) bus configuration. Disparity, if any, between the modeled and the measured results basically stem from unavoidable measurement errors for receive frames that we predicted through simulation increased our confidence in various design mechanisms that were verified using the performance model as a test platform. When designing new I/O architecture or memory implementations, our

performance model allows changes to be made easily in order to determine the impact of such changes on performance. The modeling strategy proved very effective and helped to deliver a high-quality product with better performance than what was intended initially.

Acknowledgements

I wish to acknowledge all members of the DEMFA development group for their help in modeling the adapter. Their openness to examine new designs to enhance performance resulted in this high-speed adapter. I am also grateful to the group for assisting with the performance measurements. Finally, I wish to extend special thanks to Howard Hayakawa, Gerard Koeckhoven, Satish Rege, Andy Russo, and Dick Stockdale.

References

1. R. Gillett, "Interfacing a VAX Microprocessor to a High-speed Multiprocessing Bus," Digital Technical Journal, no. 7 (August 1988): 28-46.

2. W. Hawe, R. Graham, and P. Hayden, "Fiber

3. B. Allison, "An Overview of the VAX 6200 Family of Systems," Digital Technical Journal, no. 7 (August 1988): 19-27.

4. D. Fite, Jr., T. Fossum, and D. Manley, "Design Strategy for the VAX 9000 System," Digital Technical Journal, vol. 2, no. 4 (Fall 1990):

13-24.

5. F. Ross, "FDDI-

A Tutorial," IEEE Communications Magazine, vol. 24, no. 5 (May 1986): 10-17.

6. S. Joshi, "High-Performance Networks: A Focus on the Fiber Distributed Data Interface (FDDI) Standard," IEEE MICRO (June 1986): 8-14.

7. R. Stockdale and J. Weiss, "Design of the DEC LANcontroller 400 Adapter," Digital Technical Journal, vol. 3, no. 3 (Summer 1991, this issue): 36-47.

8. S. Rege, "The Architecture and Implementation of a High-performance FDDI Adapter," Digital Technical Journal, vol. 3, no. 3 (Summer 1991, this issue): 48-63.

9. Programmer's Reference Manual for SIMULA for

Distributed Data Interface Overview," Digital Technical Journal, vol. 3, no. 2 (Spring 1991): 10-18.

VAX under VMS Operating System (North Berwick, Scotland: EASE Ltd., 1991).

10. G. Birtwistle, O. Dahl, B. Myhrhaug, and K. Nygaard, SIMULA BEGIN (Kent, England: Chartwell-Bratt Ltd., 1980).
11. L. Kleinrock, Queueing Systems, vols. 1 and 2 (New York: John Wiley and Sons, 1976).
12. D. Chiu and R. Sudama, "A Case Study of DECnet Applications and Protocol Performance," Proceedings of the ACM SIGMETRICS Conference (May 1988).
13. H. Yang, B. Spinney, and S. Towning, "FDDI Data Link Development," Digital Technical Journal, vol. 3, no. 2 (Spring 1991): 31-41.