GIGAswitch System: A High-performance Packet-switching Platform

by

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

ABSTRACT

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.

INTRODUCTION

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 make 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.

[Figure 1 (The GIGAswitch System) is not available in ASCII format.]

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

- o Implementation of protocols (Internet protocol [IP], simple network management protocol [SNMP], and IEEE 802.1d spanning tree) above the media access control (MAC) layer
- o Learning addresses in cooperation with the line cards
- o Maintaining loosely consistent line card address databases
- o 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 X 36 X 6). The crossbar is formed from three 36 X 36 X 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 $1<size<7$, 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 ~ 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

[Equation here. Please see postscript version.]

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 X 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]

[Figure 2 (Effect of Hunt Groups on Utilization) is not available in ASCII format.]

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

[Equation here. Please see postscript version.]

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.

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.

[Figure 3 (Packet Queuing on the SCP) is not available in ASCII format.]

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

[Figure 4 (Switch Control Processor) is not available in ASCII format.]

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

[Figure 5 (FDDI Line Card for Two Ports) is not available in ASCII format.]

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.

[Figure 6 (GIGAswitch System Modules) is not available in ASCII format.]

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, 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 X 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 video conferencing, 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.

TRADEMARKS

BIOGRAPHIES

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.

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.

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.

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 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.