The GIGAswitch System: A High-Performance Packet Switching Platform

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

Digital Equipment Corporation

Abstract

The GIGAswitch[1] System is a very high-performance packet-switching platform built on a 36-port 100 Mbit/second crossbar switching fabric capable of 6.25 million connections/second. The GIGAswitch System platform is data link independent. The first GIGAswitch System product offering uses 2-port FDDI line cards to construct a 22-port IEEE 802.1d FDDI bridge. A 4-port line FDDI line card is under development, as is a multifunction 2-port line card providing SONET STS-3c (155 Mbits/sec), T3 (45 Mbits/sec.) and/or E3 (34 Mbits/sec.) using ATM. Distributed forwarding, in hardware, yields forwarding rates greater than 200,000 packets per second per port. 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.

Overview

The GIGAswitch System is a very high-performance multiport packet-switching platform. The first product offering using the GIGAswitch System platform is a modular IEEE 802.1d FDDI bridge[1] with up to 22 ports. This product consists of four module types (FDDI Line Card (FGL), Switch Control Processor (SCP), Clock Card and Crossbar Module) that plug into a backplane in a 19-inch rack mountable cabinet. (See Figure 1) The power and cooling systems provide N+1 redundancy, with provision for battery operation.

The GIGAswitch System uses the combination of distributed forwarding hardware and crossbar switching to attain very high performance. When a packet is received, the receiving line card decides where to forward the packet autonomously. GIGAswitch System ports are fully interconnected with a custom VLSI crossbar that permits up to 36 simultaneous conversations. Data flows through 100 Mbit/second point-to-point connections, rather then through any shared media. Movement of unicast packets through the GIGAswitch System is accomplished completely by hardware.

The high performance of the GIGAswitch System allows it to eliminate network hierarchy and concommitant delay by collapsing backbones, to aggregate traffic from LANs, and to construct workstation farms. LAN and 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.

---

[1]GIGAswitch is a trademark of Digital Equipment Corporation

---

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

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 that is implemented in the Gigaport Interface custom VLSI chip (GPI).

Hardware Overview

Four module types in a 19-inch rack-mounted cabinet with power supplies and fans comprise the minimum GIGAswitch System A two-port FDDI line card (FGL-2) is the first line card type implemented for the GIGAswitch System; the four-port version, currently under design, is the FGL-4. This card 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 Switch Control Processor (SCP) provides a number of centralized functions, including implementation of protocols above the MAC layer (IP, SNMP and IEEE 802.1d spanning tree), learning addresses in cooperation with the line cards and maintaining loosely consistent line card address databases, forwarding multicast packets and those for which the destination is not known, switch configuration, and network management through both SNMP and the GIGAswitch System out-of-band management port. The clock card provides system clocking and storage for management parameters, while 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 making a seemingly endless series of design decisions and tradeoffs. In this section we discuss some of the major issues in the GIGAswitch System design and implementation.

Multicast

While very high packet forwarding rates for unicast packets are required to avoid being a network bottleneck, our experience with large extended LANs indicated that considerably lower rates would achieve the same result for multicast packets. 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 in a LAN with filters so that measuring extended LAN backbones yields little multicast traffic.

While the GIGAswitch System forwards unicast traffic in a distributed fashion, the multicast forwarding implementation is centralized, and a substantial portion is performed by software. The GIGAswitch System can also limit the rate of multicast traffic emitted by the switch, thus preventing lower-speed LANs attached to the switch through bridges from being swamped by high multicast traffic rates.

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

---

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

Switch Fabric

The core of the GIGAswitch System is a 100 Mbit/second full-duplex 36x36x6 non-blocking crossbar formed from 3 36x36x2 custom VLSI crossbar chips. Each crossbar input is paired with a corresponding output to form a dual-simplex datapath. The 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 fabric (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 datapath speed of 150 Mbits/sec using a 25 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 in CMOS. The FDDI data rate was used as the design center for the crossbar; higher data rates can be accommodated through the use of *hunt groups*, *cf.* below. The maximum connection rate for the crossbar depends on the switching overhead, *i.e.,* the efficiency of the crossbar output port arbitration, connection setup and teardown 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 where 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.

Except for some maintenance functions, crossbar output port arbitration is done using logical addresses. The arbitration mechanism, called *take-a-ticket,* is similar to one 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 GigaPort Interface (GPI) chips on the line cards and SCP. The GPI is a custom CMOS VLSI chip with about 85,000 transistors. Ticket and connection information are communicated among GPIs over a bus in the switch backplane. While it is necessary to use backplane bus cycles for crossbar connection setup, an explicit connection teardown is not performed, reducing the connection setup overhead. This optimization doubles the connection rate, making GIGAswitch System capable of 6.25 million connections per second.

The GPI allows many physical ports to be assigned the same logical address, creating a *hunt group*. To a sender, a hunt group appears like 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 in [2], but without restrictions on group membership. 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 do about 1.3 million lookups per second per port, 880,000 of which are for 48-bit quantities. The 48-bit lookups must

---

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

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. We used different approaches for looking up the fields in the received packet depending on the number of values that a field could have.

Current content-addressable-memory (CAM) technology currently provides about 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, and fast, expensive RAM would be needed for the lookup tables in order to minimize the forwarding latency.

To meet our lookup performance goals at reasonable cost the FDDI to GigaNet (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 4 reads from the off-chip static RAM chips that are also used for packet buffering.

Each 48 bit address is hashed using a 48 bit programmable hash multiplier. Each address is treated 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 non-zero 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 either be empty, or 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, the pointer either points to the lookup record associated with this address, or the address is not in the tables, but just happens to collide with a known address. To distinguish these cases, the remaining upper 32 bits of the hashed address is compared to the previously computed upper 32 bits of the hash of the known address stored in the lookup record. One of the properties of this hash function is that it is a one-to-one and onto mapping from the set of 48 bit values to the same set. As long as the lookup table records are not shared by different hash buckets, comparing the upper 32 bits is sufficient and leaves an additional 16 bits of information to be associated with this known address.

In the case where $1 < size \leq 7$, the pointer stored in the hash bucket points to the first entry in a balanced binary tree of depth 1, 2 or 3. This binary tree is an array sorted by the upper 32 hash remainder bits. No more than 3 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 7 addresses collide in the same hash bucket, the additional overflow addresses are stored in a content-addressible memory called the GCAM. This is a very rare occurrence. After several dozen overflow addresses have been added to the GCAM, the system decides that it has a poor choice of hash multipliers. It then initiates a re-hashing operation, whereby the SCP module picks a better 48 bit hash multiplier that it redistributes 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 non volatile memory.

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

Packet Buffering

The FDDI line card provides both input and output packet buffering for each FDDI port. Output buffering is used to buffer packets when the outgoing FDDI link is busy. Input buffering is used to buffer packets during switch arbitration for the desired destination port. Both input and output buffers are divided into separate FIFO queues for different traffic types.

Switches that have a single FIFO queue per input port are subject to a 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 that 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% 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% for two active ports. When the independent trial assumption is removed, the actual mean utilization has been computed [4] and is approximately 60% for large numbers of active ports.

Hunt groups are another factor that affects utilization. The benefits of hunt groups on head-of-line blocking can be seen by extending the simple independent-trial analysis. The estimated mean utilization is

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

where n is the number of groups and g is the hunt group size *(i.e.*, all groups are the same size in this model and the total number of switch ports is $n \times g$). This result is plotted in Figure 2 along with simulation results that remove the independent trial assumption. These results agree with those reported above for the case when there is 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 in Figure 10 of [2].

The most important factor in head-of-line blocking is the distribution of traffic within the switch. For example, in the case where all traffic is concentrated to a single output, then there is zero head-of-line blocking since traffic behind the head of the line cannot move any more easily than the head of the line. To study this effect, we extended the simple independent-trial model above to estimate utilization when the traffic from a larger set of inputs is uniformly distributed to a smaller set of outputs (for example, a larger set of workstations each sending data uniformly to a smaller set of file servers). The result is:

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

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

Another important factor in head-of-line blocking is the nature of the links and the traffic distribution on the links. Standard FDDI is a simplex link. In this case, simulation studies of a GIGAswitch System model have shown that the mean utilization of a set of standard FDDI links will approach 100% if each FDDI sources approximately 50

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

Mbits/sec of fixed-size packets into the switch with uniform destinations to all FDDI links in the set despite head-of-line blocking. The reason is that almost 50% of the FDDI bandwidth is needed to sink data from the switch and hence the switch data path is only at 50% of capacity when the FDDI links are 100% utilized. This result also applies to duplex T3 (45 Mbits/s) and all slower links. In these situations, the switch operates at well below capacity, with little internal queueing.

In summary, our studies of buffering in the GIGAswitch System have led us to conclude that head-of-line blocking will rarely be an issue for actual customer traffic and that hunt groups are effective in reducing head-of-line blocking if it does occur. Other solutions for head-of-line blocking would have raised the cost of the GIGAswitch System and benefited very few customers.

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 even when the packet forwarding loads approach the worst case maximum. The techniques used to guarantee forward progress on activities like the spanning tree include pre-allocation 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: packets to be flooded; 802.1d control packet packets; intra-box Inter Card Command (ICC) packets; and Simple Network Management Protocol (SNMP[5]) packets. 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.

In a GIGAswitch System fully populated with FGL-2 line cards, important packets may be buried in a blizzard of $> 200 kpackets / port / \sec \times 22 ports = 4.4 Mpackets / \sec$. This is too great a load for the SCP CPU to handle on its own.

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 pre-allocating 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 hardware. 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 first-in, first-out (FIFO) order, then although reception of each packet type would be ensured, 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

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

if the amount of work to do is no greater than the enqueue/dequeue overhead.) As 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 the 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 5.

At time t1, packets arriving in non-empty quotas are transferred by DMA into DRAM. 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

The discussion of packet types and buffer quotas in the previous section may have been taken in the simple sense of distinguishing important bridge-to-bridge messages from a blizzard of unknown-DA and multicast messages. Such simple distinctions would not, however, prevent a malicious station from consuming all of the buffers for multicast packets. Simple distinctions would allow starvation of protocols such as ARP that are multicast based and that become important when they are not allowed to function.

In order 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, though, 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 5 contains the layer of flooding output queues, which 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. This is compensated for in the SCP by providing shared resource pools of buffers, and by high-throughput, low-latency packet forwarding.

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, interrupt rate limiting 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.

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

Non-interrupt 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 so interrupt rate limiting alone is sufficient to limit interrupt execution time. Most error processing falls in 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. So, 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 driven can lead to the lower throughput if the CPU is sometimes idle. This can be avoided by augmenting interrupt processing with polled processing when there are idle cycles left over after all activities have had some minimal fair share.

Reliability and Availability

The GIGAswitch System minimizes the time that the network is down due to switch failures, repairs, or upgrades. We made components in the GIGAswitch System that could be single points of failure simple, and thus more reliable. Complex functions (logic and firmware) were placed on modules that could be made redundant. A second SCP, for example, will take over 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 will place one of the ports in backup state; failure of the operational port will cause 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

Clock Card

The clock card generates the system clocks for the GIGAswitch System modules and contains a number of centralized system functions. We placed some of these functions on the clock card, rather than on the backplane, because we wanted the backplane, which is difficult to replace, to be passive and thus more reliable. These functions included storing the set of 48-bit IEEE 802 addresses used by the switch and arbitration logic for the backplane bus that is used for connection setup and some inter module communication. We also took advantage of the central nature of the clock card to simplify some functions that might have been provided by the SCP. Management parameters are placed in a stable store in FLASH 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 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

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

card does not dramatically increase complexity of that module (or reduce its reliability) but significantly reduces the complexity and increases the reliability of the system as a whole.

Switch Control Processor

The SCP module contains a MIPS R3000 CPU with 64 KByte instruction and data caches, write buffer, 16 MByte DRAM, 2 MByte FLASH, and custom crossbar connection. A diagram of the SCP is presented in Figure 4. The large DRAM provides buffering for packets forwarded by the SCP and contains the switch address databases. The custom crossbar connection (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 motivation for choosing the MIPS processor was the fact that good software development and simulation tools were available.

We used simulation of the SCP software to evaluate hardware and software design tradeoffs. Address traces from simulations of the SCP software were used as input 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 tradeoffs. We learned that for compiler-generated code, loads and stores account for about half the executed instructions, so cache size and write buffer depth are important. This meant, for example, that some integrated versions of the mips R3000 processor would not perform well in our application. Avoiding stale data in the cache by triggering cache refills accounted for over one third of the data cache misses, so an efficient mechanism for getting the cache up-to-date with memory is important. However, it is not important that the cache is updated by all DMA input activity, since 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 assembler, yielding tighter code and fewer memory references.

The conventional MIPS method for forcing the cache to be updated from memory incurs three steps of overhead[6]. One step is linear in the amount of data to be accesses, 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 (2 for FGL-2, 4 for FGL-4) and a processor subsystem. The FDDI port systems (see Figure 6) are completely independent and 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 68302 microprocessor with 1 MByte of DRAM, 512KByte ROM, and 256 KByte of Flash memory that is used for initial configuration and setup, diagnostics, error logging, and firmware functions. The FGL reused much firmware from other network products. Much of the firmware was derived from the DECnis 600 FDDI line card firmware; SMT functions are provided by the Common Node Software (CNS) [7].

The FGL uses daughter cards, called ModPMDs, to implement a variety of physical media attachments, including single and multimode fiber usable for distances of up to 2 and 20 kilometers, respectively, and unshielded twisted pair copper wire (UTP), usable up to 100 meters. Both the FGL-2 and FGL-4 can support four ModPMDs. The FGL-2 can support 1 or 2 attachments per FDDI LAN and appear as a Singly Attached Station (SAS), Dual Attached Station (DAS), or M-port, depending upon the number of ModPMD cards present and management

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

settings. The FGL-4 supports only SAS configurations. The DEC VLSI FDDI chip set. was used to implement the FDDI Physical layer and MAC layer protocols. Each FDDI LAN can be either an ANSI standard 100 Mbit/sec. ring, or a full-duplex (200 Mbits/sec.) point-to-point link using Digital's Full Duplex FDDI extensions[8].

The heart of the FGL is the bridge forwarding component, which consists of two DEC designed chips: the FDDI-to-GigaPort Controller (FGC chip) and GIGAnet Content Addressable Memory (GCAM chip), plus 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 DSAP fields, and 5 byte SNAP protocol identifiers. The GCAM has two data interfaces: a 16 bit interface used by the processor and a 8 bit read only interface that is used for on-the-fly matching of packet fields. The FGC can initiate a new GCAM lookup every 80 nanoseconds.

The FGC chip is a large (~250,000 transistors), 240 pin 1.0 micron 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 whose size is determined by firmware at startup time. The FGC looks up various FDDI packet fields in order to determine whether to forward a packet, and if so on which port; to determine whether the packet contains a new source address and to note when each source address was last heard, and 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

Packet Flow

Figure 3 illustrates 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 Mbit/sec. connections to the crossbar; traffic can flow in both directions simultaneously.

The source address (SA), destination addresses (DA) and protocol type (PID, which may include the FC, DSAP, SNAP, *etc,* fields) of each packet received by a GIGAswitch System line card are looked up by the FGC hardware on the line card. 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 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 begin forwarding 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. Network management packets, multicast (and broadcast) packets and packets for which the destination address is not known (*unknown DA packet*s) are sent to the SCP. The SCP is responsible for coordinating the switch-wide resources necessary to forward multicast and unknown DA packets.

Learning and Aging

A transparent bridge *learns* the 48-bit MAC addresses of nodes in a network by receiving all packets on every LAN connected to the bridge and making note of the bridge port on which each source address was seen. The SCP

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

and line cards cooperate to build a master table of 48 bit address to port mappings on the SCP, and to 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 some 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 in a while on a list that will is retrieved at regular intervals by the SCP. Aged-out 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 there is a LAN topology change. 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, placing addresses that have been seen but are not owned by this port on a list that is retrieved by the SCP. The SCP can then quickly cause the address to be owned by the new port.

Multicast to Multiple Interfaces

Sending a packet out multiple ports is delegated to the SCP. This simplifies the line cards and provides centralized information about packet flooding so as to avoid overloading remote lower-speed LANs with flooded packets. The SCP allows rate limits to be specified for unknown destination and for multicast destination traffic to be set by network management.

Flooding a packet requires replicating the packet and transmitting the replicas on a set of ports. Design of the GIGAswitch System flooding mechanism required deciding where the replication takes place: on the SCP, in hardware or software, or on the line cards. Design criteria included how much crossbar bandwidth is consumed by the flooding, and the effect of the flooding implementation on the forwarding performance of unicast packets.

The size of the switch and the filters that are specified affect this decision. If the typical switch is fully populated with line cards and has no filters set, then one incoming multicast packet will be 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 ports.

Since the crossbar fabric 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 since less iteration of information is 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 packet is transferred by DMA once into a private memory and all iteration proceeds from that memory, removing contention at the SCP's DRAM. The

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

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

The GIGAswitch System software development and test were done in parallel with the hardware development.

The vast majority of the SCP software development effort occurred before reliable SCP hardware arrived in the lab, and before integration of the various modules could proceed in a GIGAswitch System. The SCP software was developed using a simulation that 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 whether to use line card models connected to real networks, or to use real GIGAswitch System FGLs. We were able to extract instruction and data address references from the instruction interpreter that understood the SCP memory map.

FGL testing initially proceeded in stand-alone mode without other GIGAswitch System modules. The FGL firmware provided a test mechanism that allowed Inter Card Command (ICC) packets to be received and transmitted over a serial port on the module. This facility was used to test ICC processing by sending ICCs from a host computer over a serial line. The next level of testing used pairs of FGLs in a GIGAswitch System backplane. SCP hardware was not available at this time, and one FGL ran debug code that sent ICCs through the crossbar, emulating the SCP.

The serial ICC interface to the FGL used in the first FGL testing phase was used to connect an FGL to SCP software emulated on workstations, allowing SCP and FGL firmware to communicate with ICCs. This allowed testing SCP/FGL interactions long before the SCP hardware was ready.

Network Management

The GIGAswitch System is manageable via the Simple Network Management Protocol (SNMP). 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* 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 will be 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 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 control of a workstation, these testers can send, receive and compare packets at the full FDDI line rate. We used 21 of these testers in conjunction with commercial LAN analyzers to measure GIGAswitch System performance.

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

The GIGAswitch System 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 by the fact that 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 about 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 exagerate differences in the forwarding rate. The difference between 270,000 packets/second and 227,000 packets/second is less than nine byte times per packet at 100 Mbit/sec.

The GIGAswitch System 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 measuring it that way 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 measured at about 14 microseconds, or about 175 byte times at 100 Mbits/sec. This measurement illustrates the result of applying distributed dedicated hardware to the forwarding path, as it includes two 48-bit address and a PID lookup on the incoming line card., the output filtering decision on the outbound line card, as well as the connection latency and data movement and synchronization delays.

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

By design, the GIGAswitch System flooding rate is lower than the rate for unicast traffic. With an offered load of 3000 multicast packets per second, we have measured the flooding rate at 2700 packets per second. Note that a multicast rate of r implies transmission of $(s \times r)$ packets by the switch, where s is the number of ports currently on the bridge spanning tree. Measurement of GIGAswitch System flooding rates on real networks within Digital and at field test sites indicates that the switch is not a bottleneck for multicast traffic.

Conclusions

The GIGAswitch System is a general-purpose packet switching platform that is data link independent. We have discussed many of issues and techniques that 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 6.25M variable-size packets per second includes support for large hunt groups with no loss in performance. The first GIGAswitch System product was a 22-FDDI-port IEEE 802.1d-compliant bridge that was shipped to general customers starting in June 1993. The 2-port GIGAswitch System line card that interfaces to SONET STS-3c/SDH STM-1 using ATM, or interfaces to DS-3 or E3 using ATM was announced in Oct. 1993 and is planned for shipment in May of 1994. A 4-port FDDI line card is also in development.

The GIGAswitch System can be naturally extended to include other data links such as HSSI, fast Ethernet, and Fibre Channel. Additional functions such as high-performance routing are also possible.

The GIGAswitch System has already been successfully used in a wide variety of applications areas including workstation farms, high energy physics, LAN backbones, and MAN backbones. One or more GIGAswitch Systems can also 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

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

support thousands of conventional LAN users as well as emerging applications such as large-scale video conferencing, multi-media, and distributed computing.

Acknowledgments

Our thanks to the following engineers who contributed to the GIGAswitch System architecture and product:

Vasmi Abidi, Dennis Ament, Desiree Awiszio, Juan Becerra, Dave Benson, Chuck Benz, Paul Clark, Jeff Cooper, 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, 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 Windagle

References

1. *Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges,* (New York, NY: Institute of Electrical and Electronic Engineers, Inc., IEEE Std 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. J. Gilbert, *Modern Algebra with Applications,* (New York, NY: John Wiley, 1976).

4. Mark J. Karol, Michael G. Hluchyj and Samuel P. 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)*, IETF RFC 1157

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

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

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

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

Figure 1: GIGAswitch System Photograph

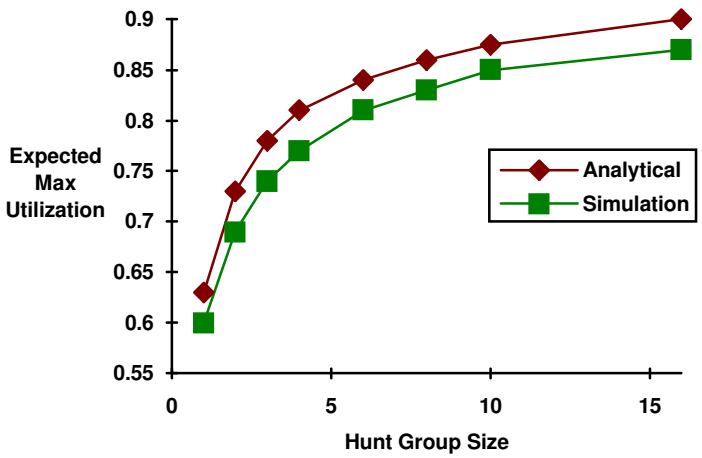

Figure 2: Effect of Hunt Groups on Utilization

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

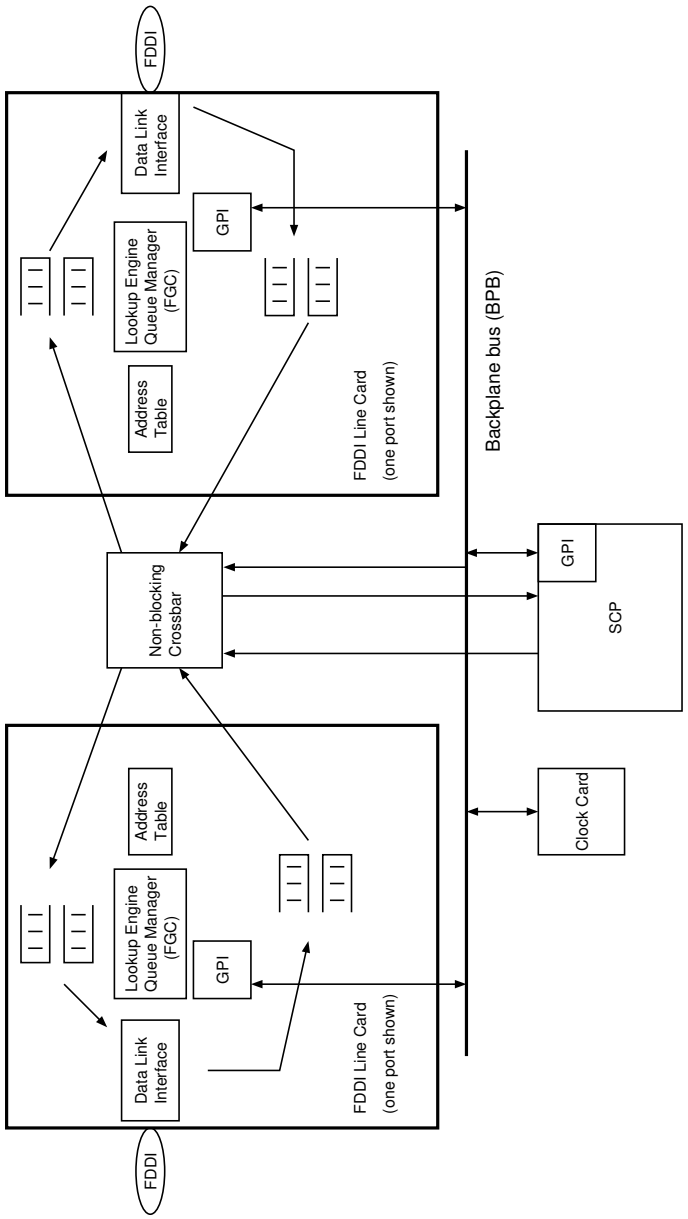

Figure 3 GIGAswitch System block diagram

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

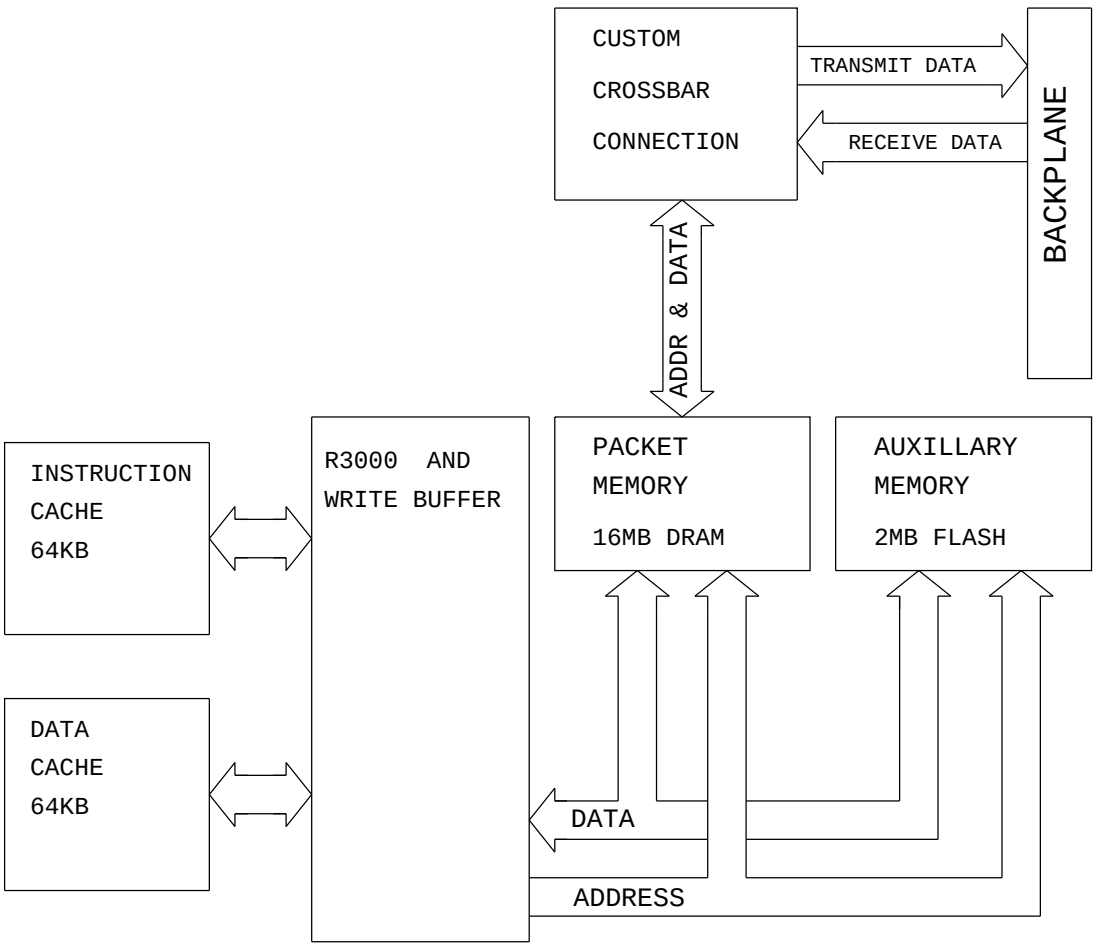

Figure 4: SCP Block Diagram

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)

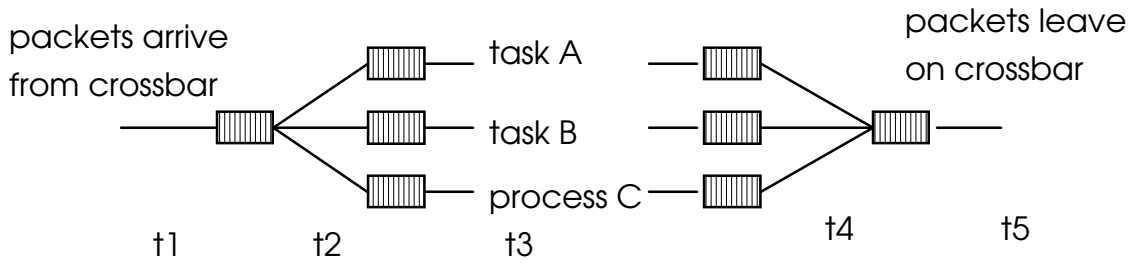

Figure 5: SCP Packet Queuing

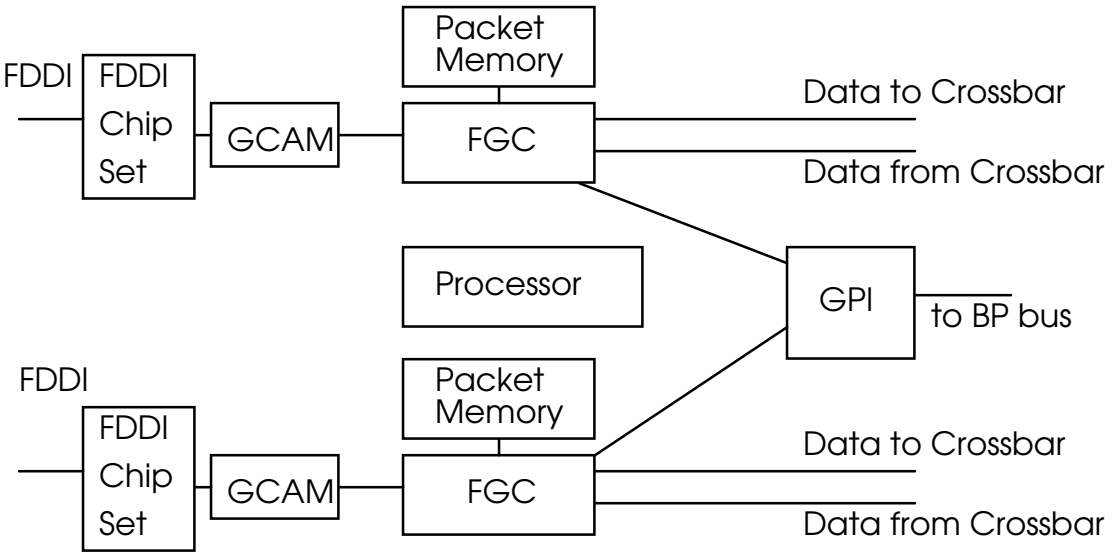

Figure 6 FGL2 Block Diagram

This is a prepublication version of a paper scheduled to appear in the *Digital Technical Journal*, vol. 6, no. 1 (1994)