# THE BAY BRIDGE: A HIGH SPEED BRIDGE/ROUTER

by

Nick McKeown, Richard Edell, and My T Le

Memorandum No. UCB/ERL M92/91

27 August 1992

# THE BAY BRIDGE: A HIGH SPEED BRIDGE/ROUTER

by

Nick McKeown, Richard Edell, and My T Le

Memorandum No. UCB/ERL M92/91

27 August 1992

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering
University of California, Berkeley
94720

# THE BAY BRIDGE: A HIGH SPEED BRIDGE/ROUTER

by

Nick McKeown, Richard Edell, and My T Le

Memorandum No. UCB/ERL M92/91

27 August 1992

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering
University of California, Berkeley
94720

# The Bay Bridge: A High Speed Bridge/Router

Presented at IFIP PfHSN Workshop, Stockholm, Sweden, May 1992.

Nick McKeown Richard Edell My T Le

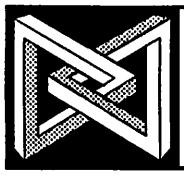
**Abstract**

The Bay Bridge is a high performance bridge/router capable of bridging and routing over 100,000 packets per second between two network ports. The first prototype provides a platform for the investigation of very high speed bridging and routing in hardware. High throughput is achieved by a specialised processor: *The Protocol Converter*, a programmable device for translating between network protocols and making forwarding/routing decisions.

The first implementation is an encapsulating two-port remote bridge with two network interfaces: FDDI rings are interconnected via the public SMDS network operating at the SONET STS-3c or DS3 rate. The first IEEE 802.6/D15 conformant DQDB MAC chip operating at STS-3c and DS3 rates has been developed as part of this project. Plug-in replacement SMDS interfaces have been designed for each rate.

The architecture is split into four main blocks: the Bridge Board, the SMDS Interface, the FDDI Interface and the Host Interface. Each architectural block is described in turn.

Finally, a schematic design is presented for a multi-port hardware router based on an ATM-switch backplane.



The Bay Bridge

Project Report: 12 Revision: 2.0
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

# 1 Glossary

**FDDI:** ANSI Fiber Distributed Data Interface.

**Frame:** Used to describe FDDI frames or packets. Maximum frame length is 4500 bytes.

**SMT:** FDDI Station Management Protocol.

**SMDS:** Switched Multimegabit Data Service, defined by Bellcore [1].

**DQDB:** Distributed Queue Dual Bus Protocol defined by IEEE 802.6 Committee. Used as the medium access control protocol for SMDS.

**Message:** Used here to describe an SMDS L3_PDU (DQDB IMPDU). Maximum message length is 9188 bytes.

**Cell:** Used here to describe an SMDS L2_PDU (DQDB DMPDU). Cells contain 44-bytes data with a 9-byte header.

**MID:** Message Identifier. A negotiated, unique 10-bit message identifier.

**CSN:** Cell Sequence Number. 4-bit value to indicate ordering of cells within a message.

**Bridge Group:** Multicast E164 Address common to all communicating bridges within a group.

# 2 Introduction

## 2.1 Objectives

The purpose of this work is:

*To investigate high performance bridging and routing in hardware.*

A platform has been built, based on a programmable bridge/router board connected to a host workstation. As examples of two high speed networks, FDDI and SMDS were chosen. However, the architecture is not limited to these network protocols.

The first implementation was designed with the following objectives:

**High Throughput.** Bridging over 100,000 packets per second. Routing is expected to approach this rate.

**SMDS rates.** SONET STS-3c 155Mbps, DS3 45Mbps.

**Expandable Address Tables.** The internal address tables contain 4096 addresses by default, but may be expanded arbitrarily.

**Flexible Configuration.** The interfaces between the Bridge Board and each network card are identical. The bridge may be reconfigured as an FDDI-FDDI bridge, or other network cards may be readily added to the architecture.

**Programmable Protocol Conversion.** A custom designed Protocol Converter is used to convert between different protocols: this is the heart of the bridging/routing function of the system. The converter may be readily programmed to handle different MAC layers and routing protocols.

## 2.2 Functional Overview

The Bay Bridge is a high-throughput two-port encapsulating remote bridge/router interconnecting FDDI rings and the public SMDS network operating at STS-3c SONET (155Mbps) [4] or DS3 (45Mbps) rate. The system complies with the proposed IEEE 802.1g Remote Bridge Draft Standard and the proposals of the IEEE 802.6 Multiport Bridge Committee [3, 2]. Hardware routing assist is achieved by the Protocol Converter. This is described in Section 3.1.2.

A typical application and topology is shown in Figure 1. FDDI frames are encapsulated into a single SMDS message and transmitted across the Subscriber Network Interface[1] to the Metropolitan Switching System (MSS). The SMDS message is then delivered to one or more remote Bay Bridges. The bridge is capable of operating at a sustained maximum throughput in excess of 100kpps. A host interface provides an FDDI and SMDS interface to a local workstation. This workstation runs the Spanning Tree Bridge Topology Protocol[2] [10, 2], the Network Management agent, the FDDI Station Management Process and enables a network manager to set local and remote addresses as well as monitor network performance parameters.

[1]The Subscriber Network Interface is the DQDB protocol of IEEE 802.6

[2]The Spanning Tree Protocol allows an arbitrary topology of remote bridges and FDDI rings. This allows two or more remote bridges to connect one local tree of FDDI rings to the public network for load-balancing or redundancy.

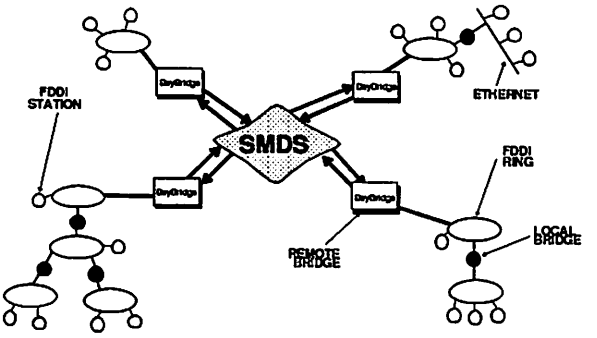



Figure 1: A typical topology interconnecting FDDI rings. Each FDDI ring may contain local bridges conforming to the IEEE 802.1d Standard.

# 3 Architecture

Figure 2 is a block diagram of the Bay Bridge architecture. The bridge consists of four main blocks: the Bridge Board, the SMDS Interface, the FDDI Interface and the Host Interface.

The interfaces between block (Port A, Port B and the Host Port) are identical but *separate* buses. This eliminates the bottleneck of a single shared system bus. Each interface consists of two unidirectional data buses and a control bus for reading and writing configuration registers. Each data bus is 32-bits wide with a "tag-bit" to delimit protocol data units, a write control line to indicate that the data is valid and a full-flag for flow control. This common interface allows the system to be configured as an FDDI-SMDS, FDDI-FDDI or SMDS-SMDS bridge/router. Alternatively, the FDDI Board or SAR Board may be connected directly to the SBus DMA card, or may be used to test each board using one or more of the SBus DMA cards.

## 3.1 Bridge Board

The Bridge Board is responsible for the forwarding and filtering of frames. This is carried out by a pair of Protocol Converters and the Address Tables.

A number of operations are carried out on all incoming frames.

1. Determine the destination port(s) of the frame.
2. Convert the frame to format of the output port(s).

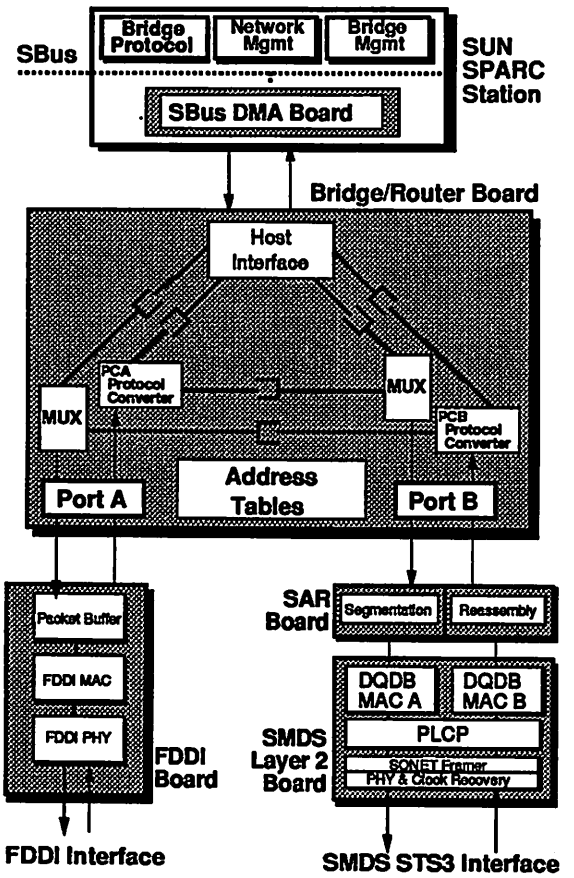



Figure 2: System Architecture showing main functional blocks. The five shaded boxes indicate each circuit board.

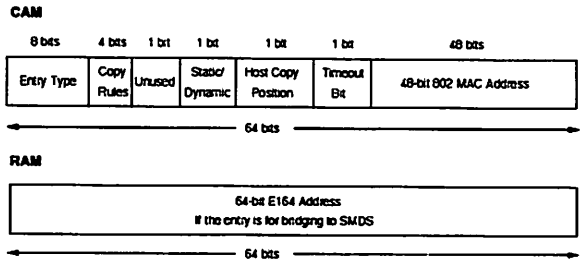



Figure 3: Format of Address Tables entries.

3. Determine the destination address of the frame.

### 3.1.1 Address Tables

The address table provides storage for 4096 addresses (expandable with a daughterboard). Each entry consists of 64 bits of content-addressable (associative) memory (CAM) and 64 bits of non-associative memory (RAM). The format of the entries is determined by the Protocol Converter microcode (i.e. the hardware doesn't restrict the format). Our current format is described in Figure 3.

The Address Tables may be accessed by either Protocol Converter or the attached workstation. Access is arbitrated on a round-robin basis. The RAM is accessed by first obtaining a match in the CAM, and then loading the index of the matching CAM entry into a memory address register.

### 3.1.2 Protocol Converter

This is a custom built, microcoded unit, optimised for fast conversion between network protocols. A block diagram is shown in Figure 4.

For a simple, flexible and fast design, a horizontal micro-instruction is used. As a result, a wide range of micro-instructions are available to the programmer for performance optimisation. The microcode is developed using a user-definable instruction set and a custom assembler. The assembled microcode is downloaded to each Protocol Converter during initialisation.

During the processing of a frame, the converter searches for patterns in protocol headers (e.g. MAC headers, LLC, SNAP, IP etc.) using the ALU and stores the frame header in the header FIFO. The Protocol Converter arbitrates for and consults the Address Tables to make forwarding and filtering decisions.

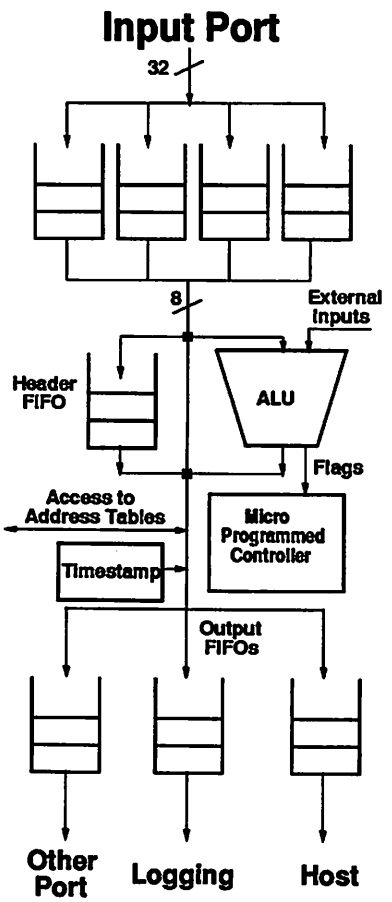



Figure 4: Block diagram of Protocol Converter: a programmable device for converting between network protocols.

When a decision has been made, the output frame is built and forwarded to one or more output channels. Frames destined for the other network interface are forwarded to the *Other Port* channel, frames destined for the host are forwarded to the *Host* channel. Logging frames may also be sent to the local host to log performance statistics or to provide call-billing information. Call logging is user defined as part of the downloaded program to the Protocol Converter. For example, for call-billing a logging frame could be sent to the host each time a frame is forwarded. The logging frame could contain a timestamp (one clock cycle resolution), the source address, destination address and frame length. For performance monitoring, an ALU register could be used to count the total number of frames forwarded. Every time the register overflows, a logging frame could be sent to the host containing an identifier and a timestamp.

We now describe how the Protocol Converter is used for (1) MAC Level Bridging and (2) Hardware Routing Assistance:

**1) MAC Level Bridging: Forwarding, Filtering and Learning**

Forwarding and Filtering:

Referring to Figure 2, assume that Port A is connected to an FDDI network interface and that Port B is connected to an SMDS interface. All FDDI frames from the ring are passed to the Bridge Board across Port A, delimited by a tag bit. Protocol Converter A (PCA) is located at the *input* to Port A. When a frame arrives, PCA reads the header one byte at a time, checking each header field as it proceeds. The first byte of an FDDI frame will contain the Frame Control (FC) field indicating whether the frame is for LLC or SMT, contains 16 or 48-bit addresses, etc. If it is an SMT frame, then it is sent to the host via the *Host* channel.

If the frame is not an SMT frame, the Protocol Converter arbitrates for and consults the Address Tables. The FDDI Destination Address (DA) is used as an index into the Address Tables. If a match occurs, the *Copy Bits* are read from the table to decide where to send the frame. The copy bits may indicate that:

1. The frame is to be copied to the host.
2. The encapsulated frame is to be copied to the SMDS interface. The SMDS message header is built by the Protocol Converter and sent to the *Other Port Channel*. The SMDS destination address is read from the *Associated Data Table* in the Address Tables. The FDDI frame is copied to the *Other Port Channel* followed by the SMDS message trailer.
3. The frame is to be copied to both ports. The SMDS message header is sent to the *Other Port Channel*, the FDDI frame is copied to both channels and the SMDS message trailer is sent to the *Other Port Channel*.

4. The frame is not to be copied to either port. This is because the frame is destined for another station on the same FDDI ring and therefore it is not forwarded.

If a match does *not* occur in the Address Tables, then the destination is unknown. A multicast SMDS frame is built and the encapsulated frame is sent to every bridge in the *Bridge Group*.

SMDS messages arrive across Port B and are received by Protocol Converter B (PCB).

Learning:
The Protocol Converter updates the Address Tables to *learn* the addresses of stations on the local and remote networks. PCA learns the source address (IEEE 48-bit format) of all passing FDDI frames and updates the Address Tables indicating that frames destined to this address should not be copied from the FDDI ring (case 4 above). When PCB decapsulates FDDI frames from SMDS messages, the source address of the remote FDDI station is stored in the Address Table, along with the associated remote bridge address (E164) (case 2 above).

**2) Hardware Routing Assistance.**
Hardware routing assistance is provided to the host by the Protocol Converter. The Address Tables are used to hold a cache of Network Layer Addresses (we shall assume here that the network layer is IP). These are added as static entries by the host. When a frame arrives the Protocol Converter consults the Address Tables to see if the IP address is known. If it is, the Protocol Converter builds the new MAC frame required by the destination port generating a new checksum if necessary. If the Network Address is *not* known or if the frame needs to be fragmented, the frame is sent to the host. The host runs the address resolution protocol and determines the destination address for the frame. After forwarding the frame, the host updates the Address Tables so that in future the Protocol Converter will route the frame in hardware.

## 3.2 SMDS Interface

The SMDS Interface consists of two circuit-boards: the SMDS Board and the Segmentation and Reassembly (SAR) Board.

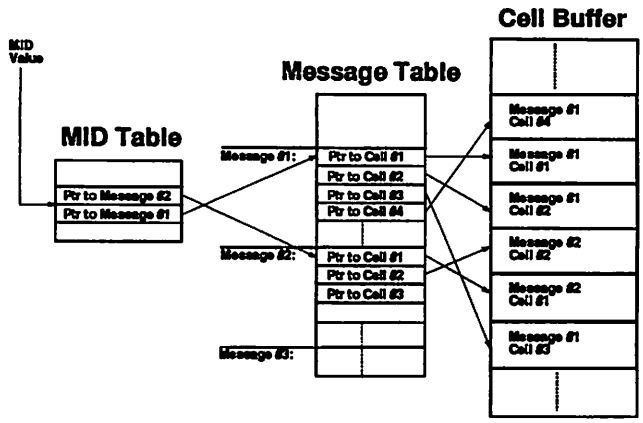



Figure 5: Reassembly for interleaved messages with misordered cells. The MID Table, Message Table and Cell Buffer are implemented as separate RAMs.

### 3.2.1 Segmentation and Reassembly Board

The SAR Board operates synchronously to the SMDS line clock, segmenting outgoing messages into cells and reassembling incoming cells into messages.

**Segmentation**
When the Bridge Board has an SMDS message to transmit over the SMDS network it hands the message to the SAR Board. Messages are segmented into 44-byte cells with 9 bytes of header and trailer information added.

**Reassembly**
Reassembly is more difficult: cells from different messages may arrive *interleaved.* The SMDS standard currently requires that cells be accepted when up to 16 interleaved messages arrive, but is expected to increase the requirement to 128 in the future. The SAR Board was designed to accept up to 128 interleaved messages. In addition and as an experiment, the SAR Board was designed to accept *mis-ordered* cells[3].

The problem is to design the reassembly buffer management efficiently. The simplest and most inefficient policy is to always allocate a buffer equal to the length of a maximum SMDS message when a BOM or SSM cell arrives. The most complex scheme is to allocate buffers of exactly the size of the SMDS message. But there is little time to make complex memory management decisions when cells arrive every $2.8\mu s$.

[3]Mis-ordered cells are normally rejected in SMDS — this is an experiment in the design of fast reassembly with cell-misordering

The algorithm used in this design is based on [9] and is summarised in Figure 5. Efficient use of the cell buffer is achieved by allocating buffers cell by cell. When a BOM or SSM cell arrives, a new entry is made in the *MID Table* and a new *Message Table* entry is removed from the free list. The cell is placed in the next free location in the *Cell Buffer* and its address stored in the Message Table. When COM and EOM cells arrive, the MID is used to index into the MID Table to find the correct Message Table entry for this MID value. The cell is placed in the next free location in the Cell Buffer and its address stored in the Message Table. The position of the address in the Message Table is calculated using the Cell Sequence Number.

Reassembled messages are removed from the cell buffer by indexing through the Message Table and reading from the Cell Buffer. In this design, completed messages may be read from the cell buffer at the cell arrival rate.

The advantages of this design are:

- The cell buffer is used efficiently. This becomes more important as the cell buffer size increases. This scheme is most appropriate for a switch or for a CPE serving a large number of users.
- Calculating the address of the next cell position in the cell buffer is simple — it is the next location removed from the free list. As a result new cells may be accepted continuously back-to-back, unless the Message Table or Cell Buffer is full.

This scheme is only worth using if the size of the Message Table is a small proportion of the size of the Cell Buffer. This may not be the case in some applications. The Cell Buffer size is chosen as a fraction, $m$ of the total maximum number of interleaved messages, $L3 \times I$ bytes where $L3 = 9188$ (maximum message length) and $I = 128$ (maximum number of interleaved messages). The Message Table size in bytes is chosen as a multiple, $n$ of the number of bytes required to keep track of the maximum number of interleaved messages, $(I\frac{L3}{L2})(\frac{1}{8}\log_2(mI\frac{L3}{L2}))$ where $L2 = 44$ bytes per cell. In general, $m < 1$ assuming that the average message of length is less than the maximum and $n > 1$ so that a large number of small messages will not fill the Message Table. Figure 6 compares the required Message Table and Cell Buffer sizes for different values of $m$ and $n$. In our design, we selected $n = 1$, $m = 0.25$ to reduce the size of the fast (and expensive) Cell Buffer memory.

### 3.2.2 SMDS Board

The main features of the SMDS Layer 2 Board are:

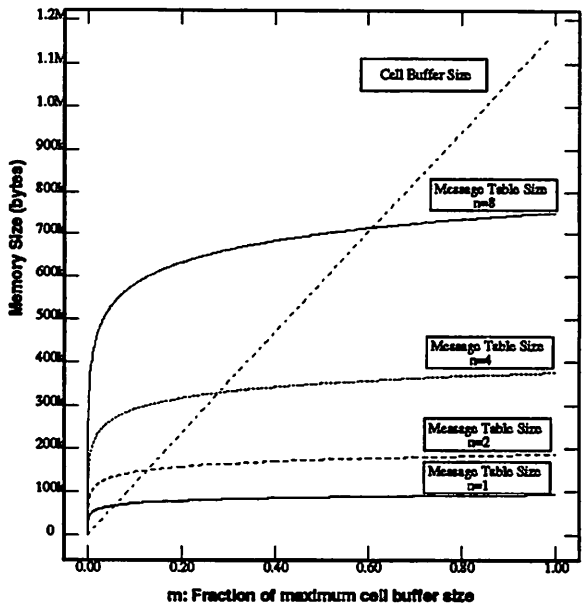



Figure 6: Comparison of the size of the reassembly Cell Buffer and Message Table. $m$ is the fraction of the total Cell Buffer size required to hold 128 maximum length interleaved SMDS messages. $n$ is the multiple of the Message Table entries required to address the cells of each message in the cell buffer.

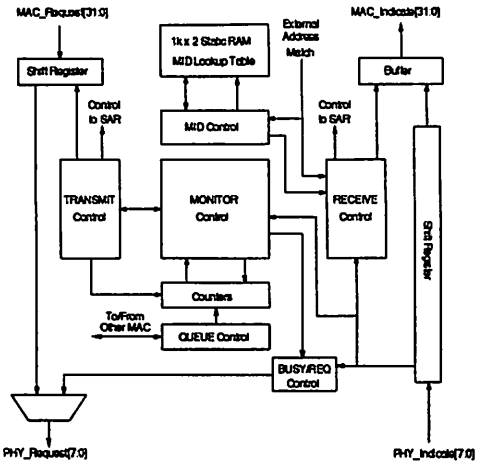



Figure 7: Block diagram of DQDB MAC chip. Designed using Berkeley VLSI design tools "Octtools" and "Lager". Fabricated through MOSIS in $2\mu m$ CMOS.

- Operates at the SONET STS-3c line rate. A replacement board operates at the DS3 rate.
- Supports Single CPE attachment to an SMDS network.
- Uses DQDB MAC chip developed as part of this project [8]. A block diagram of the MAC chip is shown in figure 7.
- Processes cells in both directions at the maximum line rate.

Figure 8 illustrates the framing format for SMDS operating at SONET STS-3c.

**Outgoing cells:**
Cells are handed to the SMDS Board by the SAR Board. When the SAR Board requests a transmission, the DQDB MAC will wait for an appropriate empty cell. When access is granted, the SAR Board passes a cell to the SMDS board. The CRC checksum is calculated by the MAC and added to the cell trailer. The cell is placed in a SONET frame by the PLCP and transmitted onto the medium by the Physical Layer.

**Incoming cells:**
Cells received from the Physical Layer are first removed from the SONET or DS3 frame structure by the Physical Layer Convergence Protocol[5] (PLCP) and handed to the MAC. The DQDB MAC accepts all incoming BOM (Beginning of Message) cells that match any external MAC address[4]; or cells with Message

[4]Default number of external addresses is 1024

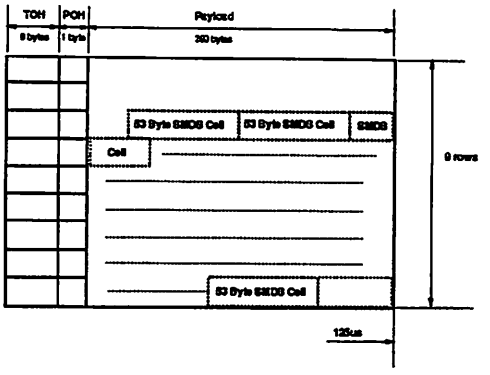



Figure 8: Proposed mapping of SMDS Cells into a SONET STS-3c frame [7].

Identifiers (MIDs) corresponding to messages in progress. The DQDB MAC maintains a table of current MIDs and alerts the Host of exceptions. The MAC does not check for correct cell ordering as this is handled by the SAR Board. The MAC checks the CRC of all incoming cells and hands cells directly to the SAR Board for reassembly, indicating if the cell has passed the CRC.

The PLCP:
For outgoing cells, the Physical Layer Convergence Protocol (PLCP) is responsible for accepting cells from the DQDB MAC, packing them into a SONET STS-3c frame structure. Cell payloads are scrambled according to the polynomial: $x^{43} + 1$.

Incoming cells are detected using the Header Check Sequence method. The PLCP searches for cell structure within the SONET frame payload. After receiving 7 correctly formed cells, the PLCP locks on the Sync Sequence and begins to transmit cells to the MAC. Before cells are handed to the MAC, they will descrambled.

The PLCP also performs the MID allocation protocol [5] using byte M2 in the *Payload Overhead* column. In addition, the PLCP records and reports other error conditions from the physical network such as: Loss of Pointer, Loss of Frame, and Far End Receive Failure to the Host.

## 3.3 FDDI Board

The FDDI board is a standard design and is not described in detail here. We use the AMD FDDI Supernet chipset to control access to the FDDI ring and a

64Kword packet buffer between the FDDI MAC and the Bridge Board.

### 3.4 Host and SBus DMA Card

The Host is a Sun SPARC Station and acts primarily as a monitor and manager; it is *not* involved in bridging data packets between the two networks.

The Host is responsible for:

**Initialising the bridge.** This involves resetting cards and device drivers, initialising statistics counters and allocating addresses. User-definable parameters, such as bridge group addresses, may be set locally by the network manager.

**Running FDDI SMT.**

**Monitoring and Statistics Gathering.** The Host monitors the performance of each board collecting information on: the traffic on each network, packet errors, address conversion hit ratio, etc. This information is presented to the network manager via the local network management agent.

**Running Network Management** The Host runs an SNMP agent to allow remote monitoring of the bridge.

## 4 System Performance

Bridging performance is usually described in bridged frames per second. The overhead is dominated by the time to process headers and not by copying data. The throughput is described here as the number of zero length frames copied per second. In particular, the bridging of packets from FDDI to SMDS is described — this is slower than from SMDS to FDDI because of the time to build the SMDS header.

The time to bridge one frame may be broken down into the following components:

**Protocol Converter Set-Up** 33 cycles.

**Address Lookup** 30 cycles.

**Address Learning** 30 cycles.

**Building SMDS Header** 45 cycles.

**Copying FDDI frame** 1 cycle per byte.

where one cycle is 50ns.

This corresponds to bridging 145,000 frames per second per port. Both ports may bridge at this rate concurrently except when they are contending for the address tables. But the address tables are used for only 60 cycles ($3\mu s$) per frame and therefore do *not* limit the per port throughput.

We may compare this with the maximum rate at which encapsulated minimum length FDDI frames may be transmitted by the SMDS interface. Minimum length FDDI frames occupy two SMDS cells each of which may be transmitted every $2.8\mu s$. This correpsonds to a throughput of 178,000 frames per second.

The throughput for routing will be lower than this, but is as yet unknown.

# 5 Technology

The DQDB MAC chip was designed using Berkeley VLSI design tools "Octtools" and "Lager". This chip was fabricated through MOSIS in $2\mu m$ CMOS. The SONET framer is from Bellcore. The SBus interface board uses an ASIC developed by Sun Microsystems for their internal use. The remainder of the project is built as five surface mounted printed circuit boards using commercially available integrated circuits (RAMs, IDT FIFOs, MUSIC Semiconductors LANCAM, standard logic components, AMD MACH programmable logic devices, AMD FDDI FORMAC Plus). Printed circuit board schematics were captured and simulated using Viewlogic's Workview software. Complex componets were simulated using VHDL.

# 6 Follow-on Work

Limitations of the current prototype include: only two ports, address table contention, and limited integration. We propose to continue this work by building a bridge/router using an ATM switch as its backplane. Requirements for the switch would include: high throughput and a simple line card interface. If no such switch is available, we plan to build one. If we build one, it would have a simple architecture, trading complexity for throughput (e.g. a crosspoint switch, such as in figure 9).

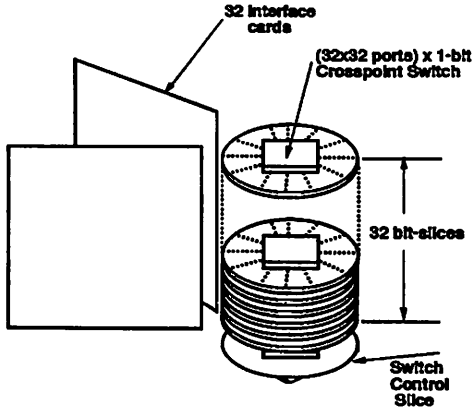



Figure 9: Architecture of simple, high bandwidth 32-bit by 32-port crosspoint switch.

Peripheral cards for the switch would include:

**Bridge/Router Interface Card** Protocol Converter(s) on the card would interface directly to the switch, and to a local address table. "Hits" in the address table would be handled locally. "Misses"' in the address table would be handled by other peripherals connected to the switch (e.g. a shared hardware address table, or a CPU). Additionally, these other peripherals would have access to update the local address tables.

**Hardware Address Table Card** This card would handle misses by other peripherals, by possibly consulting other peripherals. Thus creating a hierarchy of address tables analogous to multi-level caching. This overcomes the address table contention problem of the current prototype.

**CPU Card** The CPU card would ultimately handle address table misses. We would also use it for ATM switch control/billing. While not the objective of the proposed follow-on work, this CPU card could be used for signal processing, general purpose multi-processing, and as a disk interface providing file service to hosts attached to the ATM network (e.g. other CPU cards, hosts with ATM interfaces, or hosts connected via Bridge/Router Interface Cards).

# 7 Acknowledgements

This work was supported in part by Pacific Bell and California State MICRO Program. We would like to thank Pacific Bell for their continued support and encouragement in this work. We are also grateful to Bellcore, Sun Microsystems, Advanced Micro Devices, TranSwitch, Integrated Device Technology, Viewlogic, Hewlett-Packard and MUSIC Semiconductors for their material and technical assistance. Much credit is also due to the following students at UC Berkeley who have contributed many hard and long hours to this project: Ting Kao, Fred Burghardt, Malik Audeh, George Kesidis, Karl Petty, Steve McCanne and Nina Taft. And finally, we thank our faculty advisers Professors Pravin Varaiya and Jean Walrand for their support.

# References

[1] "Generic Systems Requirements in Support of Switched Multi-megabit Data Service"
*Bellcore Technical Reference TR-TSV-000772*, (May 91)

[2] "P802.1d MAC Bridges Draft Standard"
*IEEE Project 802 Committee P802.1d/D8*, (March 89)

[3] "P802.1g MAC Remote Bridge Draft Standard" *IEEE Project 802 Committee P802.1g*,

[4] "Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria"
*Bellcore Technical Advisory TA-NWT-000253*, Issue 6, (Sept 1990)

[5] "P802.6 DQDB Subnetwork of a Metropolitan Area Network Draft Standard"
*IEEE Project 802 Committee P802.6/D15*, (Oct 90)

[6] Robe TJ, Walsh KA
"A SONET STS-3C User Network Interface Integrated Circuit." *IEEE Journal On Selected Areas In Communications; V9 N5:732-740*, (Jun 1991)

[7] "Proposed 802.6 SONET PLCP Mapping"
*IEEE Project 802 Committee P802.6_SONET_PLCP/D3*, (Sept 90)

[8] Burghardt, Fred
"DQDB MAC Chip Datasheet"
*Bay Bridge Project Report 10, Rev 1.0*, (Jan 1992)

[9] McKeown, Nick
"Segmentation and Reassembly for Cell based Mis-ordering Networks"
Hewlett-Packard Technical Report HPL-91-176.

[10] Taft, Nina
"Bridge Protocols for an SMDS-FDDI Remote Bridge"
*Bay Bridge Project Report 5*, (June 1991)