TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

institute of telecommunications

Lecture Notes

Technical Realization of Communication Networks

Course Number: 388.031

Part 2: Interconnects and Systems

Univ. Ass. Dr. Slavisa Aleksic

Institute of Broadband Communications (E388)

Vienna University of Technology

Contents

INTERCONNECTS **5**

1.1 Introduction 5

1.2 Peripheral Component Interconnect (PCI) 6

1.2.1 Conventional PCI Bus 6

1.2.2 PCI Extended (PCI-X) 15

1.2.3 PCI Express 16

1.3 Small Computer System Interface (SCSI) 24

1.3.1 Parallel SCSI 30

1.3.2 Serial SCSI (SAS) 35

1.3.3 Internet SCSI (iSCSI) 40

1.4 HyperTransport (HT) 43

1.5 Rapid I/O 55

1.6 InfiniBand 64

1.7 Common Switch Interface (CSIX) 71

1.8 Backplanes 79

1.8.1 Classical Backplane Solutions 79

1.8.2 High-Capacity Backplanes 82

1.8.3 Optical Backplanes 85

TECHNOLOGY SPECIFIC SYSTEMS **93**

2.1 Introduction 93

2.2 SONET/SDH Systems 93

2.2.1 SONET/SDH Basics 93

2.2.2 SONET/SDH Interfaces 101

2.2.3 Case Study: SONET/SDH Framer 122

2.2.4 Case Study: Implementation of SPI-4 Interface 131

2.3 Ethernet Systems 148

2.3.1 Ethernet Basics 148

2.3.2 Ethernet Physical Layer 150

2.3.3 Medium Access Control (MAC) Layer 172

2.3.4 Logical Link Control Sublayer 176

2.3.5 Ethernet Interfaces 180

2.3.6 Case Study: 8B/10B Encoder/Decoder Implementation 193

2.4 Asynchronous Transfer Mode (ATM) Systems 202

2.4.1 ATM Basics 202

2.4.2 ATM Interfaces 206

2.5 Fibre Channel (FC) Systems 223

2.5.1 Fibre Channel Layers 223

2.5.2 Fibre Channel Topologies 225

2.5.3 Media Options 230

2.5.4 Transmission Hierarchy 231

2.5.5 Flow Control 234

2.5.6 Classes of Service 236

2.5.7 Addressing 239

2.5.8 Login 240

2.5.9 FC Components 241

LIST OF FIGURES **243**

LIST OF TABLES **251**

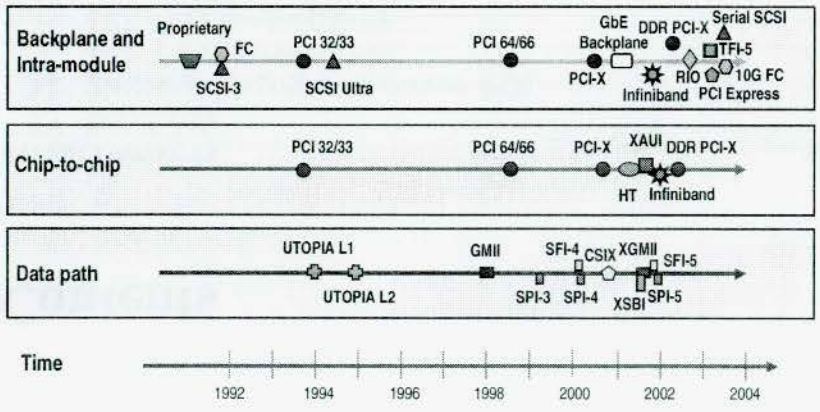
LIST OF ABBREVIATIONS **253**

Chapter 1

INTERCONNECTS

1.1 Introduction

In general, interconnects can be divided into four main categories: intra-module (backplane), chip-to-chip, and data path (in-chip) interconnects. This classification is not always straightforward because some of technologies are developed to use in several applications. Figure 1 shows an overview of the most used interfaces and interconnects.



PCI: Peripheral Component Interconnect
SPI: System Packet Interface
HT: Hyper Transport
XAUI: 10 Gigabit Attachment Unit Interface
XGMII: 10 Gbit Media Independent Interface
RIO: Papid I/O
CSIX: Common Switch Interface
SFI: SERDES Framer Interface
TFI: TDM Fabric Interface
XSBI: 10 Gigabit Serial Bus Interface
DDR: Double Data Rate

Figure 1: Interconnecting Technologies

It can be seen that, especially in the last few years, the interconnecting landscape has grown. There are a number of new technologies and standards. Most of them will be treated in this course, whereas those used for chip-to chip and backplane connections are subject of this chapter and those mostly used in technology-specific interfaces of the data path are addressed in Chapter 2.

1.2 Peripheral Component Interconnect (PCI)

The Peripheral Component Interconnect (PCI) bus, which was first introduced in the early 90's, is the dominant bus, used in both desktop and server machines for attaching I/O peripherals to the CPU/memory complex. The most common configuration of the PCI bus is a 32-bit 33 MHz version that provides a bandwidth of 133 MByte per second, although the 2.2 version of the specification allows for a 64-bit version at 33 MHz for a bandwidth of 266 MByte per second and even a 64-bit 66 MHz version for a bandwidth of 533 MByte per second. Even today's powerful desktop machines have lots of capacity available with the PCI bus in the typical configuration, but server machines are starting to hit the upper limits of the shared bus architecture. The availability of multiport Gigabit Ethernet network interface cards (NICs), along with one or more Fibre Channel I/O controllers can easily consume even the highest 64-bit, 66 MHz version of the PCI bus.

1.2.1 Conventional PCI Bus

The standard 32-bit 33 MHz PCI bus has a maximum bandwidth of 133 MByte/sec. That's shared for all devices connected to the PCI bus, including a SCSI PCI card and any other installed PCI cards. Server/Workstation boards come with multiple and higher-bandwidth PCI buses.

PCI Options:

- PCI-32/33 (133 MBye/s)
- PCI-32/66 (266 MByte/s)
- PCI-64/33 (266 MByte/s)
- PCI-64/66 (533 MByte/s)

- PCI-X 1.0
- PCI-X 66 MHz
- PCI-X 133 MHz
- PCI-X 2.0
- PCI-X 266 MHz, DDR (up to 2.13 GByte/s)
- PCI-X 533 MHz, QDR (up to 4.26 GByte/s)
- Additional features: error coding codes (ECC)
- Presently in development
- PCI-X 1066 MHz, DDR (up to 8.52 GByte/s)

It may be wise to wait for Serial ATA chipsets and motherboards to become common (mid-2003) before doing a major upgrade to your HDD and system.

Each of these additional specifications rely on the PCI spec., normally only the mechanical (form factor) definition changes. Unlike earlier PC buses, the PCI bus is processor independent.

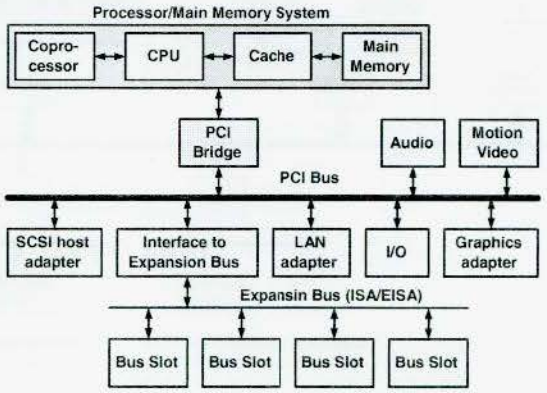


Figure 2: Block diagram of a PCI bus system

The (64bit) PCI bus is made up of the following (major) signals:

- Address/Data Bus: 64bit Address; 64bit Data, Time Multiplexed
- System Bus: 2bits; Clock/Reset
- Interface Control Bus: 7bits; Ready, Acknowledge, Stop.
- Parity Bus: 2 bits, 1 for the 32 LSBs and 1 for the 32 MSB
- Errors Bus: 2 bits, 1 for Parity and 1 for System

- Command/Byte Enable: 8 bits (0-3 at 32-bit, and 4-7at 64-bit Bus)
- 64MHz Control: 6 bits; (2) Enable/Running, (2) Present, (2) Ack/Req
- Cache: 2 Bits
- Interrupt bus: 4 bits
- JTAG Bus: 5 bits
- Power: +5, +3.3, +12, -12v, GND

A detailed list of PCI signals follows:

- ACK64# : acknowledge 64-bit transfer
- AD31-AD0: 32 address and data pins form the multiplexed PCI address and data bus
- C/BE3#-C/BE0#: command, byte enable
- CLK: PCI clock signal
- DEVSEL#: device select
- FRAME: Frame delineation
- GNT#: grant – indicates that permission to use PCI bus is granted
- IDSEL: device select during configuration (initialization device select)
- INTA#, INTB#, INTC#, INTD#: interrupt signals
- IRDY#: initiator ready
- LOCK: defines an atomic access. Used to manage resource locked on the PCI bus
- PAR: even parity for AD31-AD0 and C/BE3#-C/BE0#
- PERR#: parity error
- PRSNT1#, PRSNT2#: indicate that an adapter is installed
- REQ#: request signal to the bus arbitration unit (requests a PCI transfer)
- REQ64#: 64-bit transfer request
- RST: resets all PCI units
- SBO#: snoop backoff, indicates a hit to a modified cache line
- SDONE: snoop done
- SERR#: system error
- STOP: target-abort. Requests the master to stop the current transfer cycle.
- TCK, TDI, TDO, TMS, TRST#: JTAG boundary scan test signals
- TRDY: target ready
- 64-bit expansion

- AD63-AD32: 32 address and data pins form the expansion of the multiplexed PCI address and data bus
- C/BE7#-C/BE4#:command and byte-enable
- PAR64: even parity for the 64-bit expansion

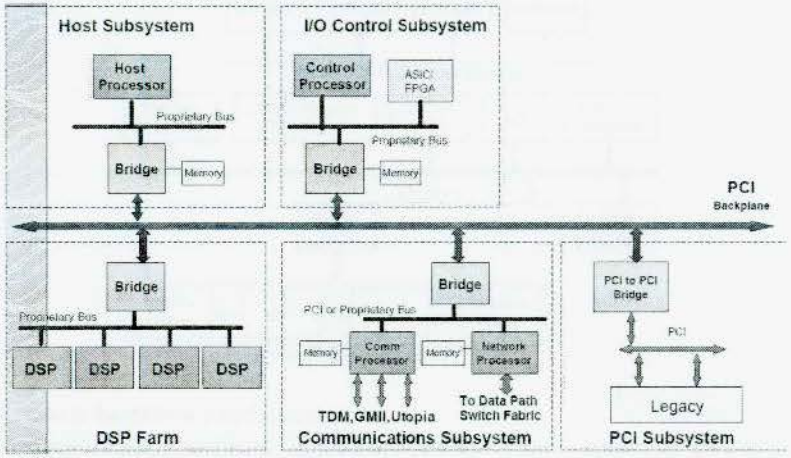
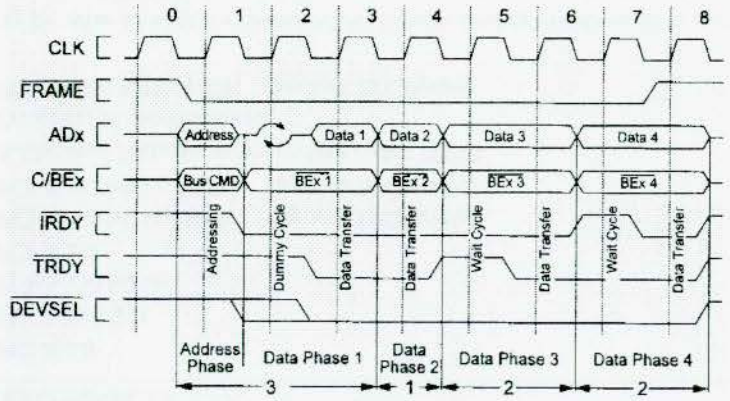


Figure 3: PCI in a communication system



AD31-AD0: 32 address and data bus
IRDY#: initiator ready
DEVSEL#: device select

C/BE3#-C/BE0#: command and byte-enable
TRDY: target ready

Figure 4: The PCI read transfer burst

The Time Multiplexed Address and Data bus may exist as either 0 to 31 bits (32 bits) or 0 to 63 bits (64 bits) using the 64-bit expansion bus. Both the Address and Data line use the same bus, Address first then Data. 32 bit PCI may also use 64-bit addressing by using two address cycles; termed Dual Address Cycles (DAC), the low order address is sent first. Additional control bits are utilized once the bus is increased to 64 bits.

The specification defines both a Reset line and a Clock line. The Clock may be either 33MHz or 66 MHz. A number of "Handshake" lines exist to allow communication, i.e. Ready, and Acknowledge

Two Parity lines are made available, one for the 32 bit bus width (bits 0 to 31) and an additional one for the 64 bit expansion (bits 32 to 63).

The following is a cycle by cycle description of the read transaction:

- Cycle 1 - The bus is idle.
- Cycle 2 - The initiator asserts a valid address and places a read command on the C/BE# signals. This is the address phase.
- Cycle 3 - The initiator tri-states the address in preparation for the target driving read data. The initiator now drives valid byte enable information on the C/BE# signals. The initiator asserts IRDY# low indicating it is ready to capture read data. The target asserts DEVSEL# low (in this cycle or the next) as an acknowledgment it has positively decoded the address. The target drives TRDY# high indicating it is not yet providing valid read data.
- Cycle 4 - The target provides valid data and asserts TRDY# low indicating to the initiator that data is valid. IRDY# and TRDY# are both low during this cycle causing a data transfer to take place. The initiator captures the data. This is the first data phase.
- Cycle 5 - The target de-asserts TRDY# high indicating it needs more time to prepare the next data transfer.
- Cycle 6 - The second data phase occurs as both IRDY# and TRDY# are low. The initiator captures the data provided by the target.
- Cycle 7 - The target provides valid data for the third data phase, but the initiator indicates it is not ready by de-asserting IRDY# high.
- Cycle 8 - The initiator re-asserts IRDY# low to complete the third data phase. The initiator captures the data provided by the target. The

initiator drives FRAME# high indicating this is the final data phase (master termination).

- Cycle 9 - FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

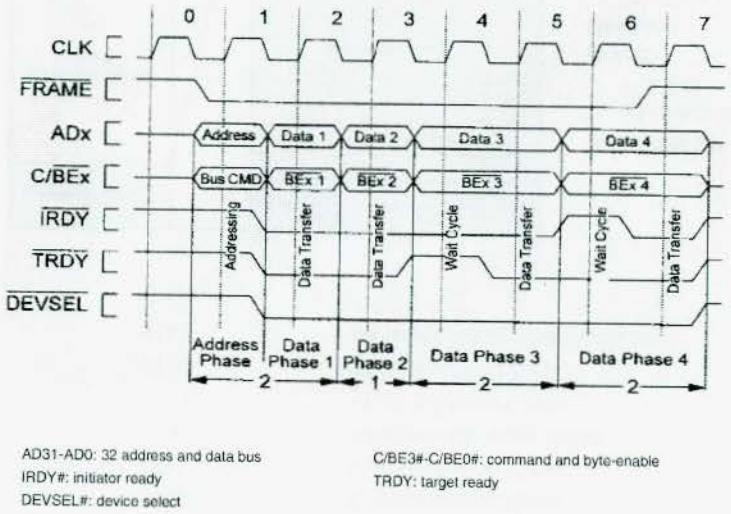


Figure 5: The PCI write transfer burst

The write transaction can be described as follows:

- Cycle 1 - The bus is idle.
- Cycle 2 - The initiator asserts a valid address and places a write command on the C/BE# signals. This is the address phase.
- Cycle 3 - The initiator drives valid write data and byte enable signals. The initiator asserts IRDY# low indicating valid write data is available. The target asserts DEVSEL# low as an acknowledgment it has positively decoded the address (the target may not assert TRDY# before DEVSEL#). The target drives TRDY# low indicating it is ready to capture data. The first data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
- Cycle 4 - The initiator provides new data and byte enables. The second data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.

- Cycle 5 - The initiator de-asserts IRDY# indicating it is not ready to provide the next data. The target de-asserts TRDY# indicating it is not ready to capture the next data.
- Cycle 6 - The initiator provides the next valid data and asserts IRDY# low. The initiator drives FRAME# high indicating this is the final data phase (master termination). The target is still not ready and keeps TRDY# high.
- Cycle 7 - The target is still not ready and keeps TRDY# high.
- Cycle 8 - The target becomes ready and asserts TRDY# low. The third data phase occurs as both IRDY# and TRDY# are low. The target captures the write data.
- Cycle 9 - FRAME#, AD, and C/BE# are tri-stated, as IRDY#, TRDY#, and DEVSEL# are driven inactive high for one cycle prior to being tri-stated.

The PCI bus treats all transfers as a burst operation. Each cycle begins with an address phase followed by one or more data phases. Data phases may repeat indefinitely, but are limited by a timer that defines the maximum amount of time that the PCI device may control the bus. This timer is set by the CPU as part of the configuration space. Each device has its own timer. The same lines are used for address and data. The command lines are also used for byte enable lines. This is done to reduce the overall number of pins on the PCI connector.

The Command lines (C/BE3 to C/BE0) indicate the type of bus transfer during the address phase. PCI was designed as a plug-and-play, self-configuring system. In support of this it defines an area of addressable ROM and RAM called configuration space, which can be interrogated to obtain information about a device and written to in order to configure it.

Each PCI device has a block of 256 bytes of configuration space: 16 32-bit double words of header information plus 48 double words of device-specific configuration registers. The header contains a vendor ID and type of device code, flags which show whether the device generates interrupts, whether the device is 66MHz-capable and other low level performance-related information, the base address locations of I/O ports, RAM and expansion ROM, the maximum latency register (mentioned earlier) and other similar general information. ROMs can contain code for different processor architectures, and a configuration register shows which ones are supported.

C/BE	Command Type
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	reserved
0101	reserved
0110	Memory Read
0111	Memory Write
1000	reserved
1001	reserved
1010	Configuration Read
1011	Configuration Write
1100	Multiple Memory Read
1101	Dual Address Cycle
1110	Memory-Read Line
1111	Memory Write and Invalidate

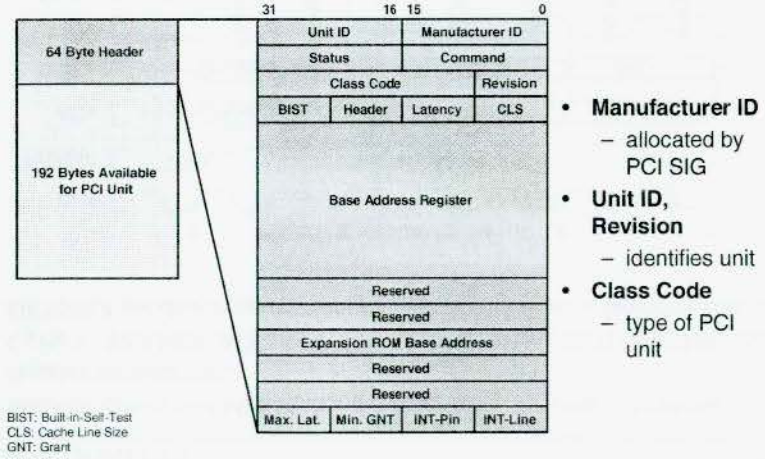


Figure 6: Configuration address space

Configuration space is completely separate from memory and I/O space, and can only be accessed using the PCI bus Configuration Read and Write commands. Intel x86 processors cannot access configuration space directly, so the PCI specification defines two methods by which this can be achieved. The preferred method, used by current implementations, is to write the target address to the 32-bit I/O port at 0CFBh, and then read or write the double word through I/O port 0CFCh.

The PCI configuration space consists of up to six 32-bit base address registers for each device. These registers provide both size and data type information. System firmware assigns base addresses in the PCI address domain to these registers.

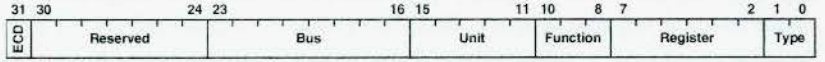


Figure 7: Accessing - address space configuration

Each addressable region can be either memory or I/O space. The value contained in bit 0 of the base address register identifies the type. A value of 0 in bit 0 indicates a memory space and a value of 1 indicates an I/O space. The following figure shows three base address registers: one for memory, one for I/O, and one for the expansion ROM types.

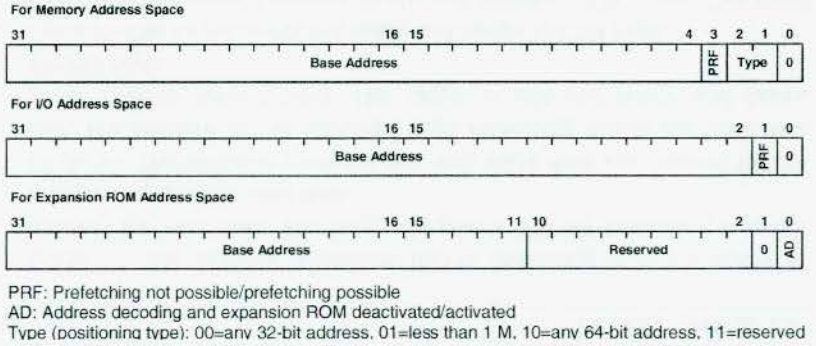


Figure 8: Base address registers

- PCI Memory Address Space

PCI supports both 32-bit and 64-bit addresses for memory space. System firmware assigns regions of memory space in the PCI address domain to PCI peripherals. The base address of a region is stored in the base address register of the device's PCI configuration space. The size of each region must be a power of two, and the assigned base address must be aligned on a boundary equal to the size of the region. Device addresses in memory space are memory-mapped into the host address domain so that data access to any device can be performed by the processor's native load or store instructions.

- PCI I/O Address Space

PCI supports 32-bit I/O space. I/O space can be accessed differently on different platforms. Processors with special I/O instructions, like the Intel processor family, access the I/O space with in and out instructions. Machines without special I/O instructions will map to the address locations corresponding to the PCI host bridge in the host address domain. When the processor accesses the memory-mapped addresses, an I/O request will be sent to the PCI host bridge, which then translates the addresses into I/O cycles and puts them on the PCI bus. Memory-mapped I/O is performed by the native load/store instructions of the processor.

1.2.2 PCI Extended (PCI-X)

PCI-X 1.0 runs at up to 133 Mhz at either 32-bit or 64-bit widths. It brings throughput up to over 8 Gbits/s. The PCI-X 2.0 standard introduces two new speed grades: PCI-X 266 and PCI-X 533. These speed grades offer bandwidths that are two times and four times that of PCI-X 133 – ultimately providing bandwidths that are more than 32 times faster than the original version of PCI that was introduced eight years ago. It achieves the additional performance via time-proven DDR (Double Data Rate) and QDR (Quad Data Rate) techniques that transmit data at either 2-times or 4-times the base clock frequency.

Because PCI-X 2.0 preserves so many elements from previous generations of PCI it is the beneficiary of a tremendous amount of prior development work. The operating systems, connector, device drivers, form factor, protocols, BIOS, electrical signaling, BFM (bus functional model), and other original PCI elements, are all heavily leveraged in the PCI-X 2.0 specification. In fact

many of these elements remain identical in PCI-X 2.0. These similarities make implementation easy.

However, the problem with the shared bus media is still present. For example, a PCI-X 1.0 system that is running at 133 MHz can have only one slot on the bus, two PCI-X slots would allow a maximum clock rate of 100 MHz, whereas the four slot configuration would drop down to a clock rate of 6 MHz. So, despite the temporary resolution of the PCI bandwidth limitation through these new upgrade technologies, there is a long term solution needed that cannot rely on shared bus architecture.

1.2.3 PCI Express

Recent advances in high-speed, low-pin-count, point-to-point technologies offer an attractive alternative for major bandwidth improvements. A PCI Express multi-drop, parallel bus topology contains a Host Bridge and several endpoints (the I/O devices).

Multiple point-to-point connections introduce a new element, the switch, into the I/O system topology. The switch replaces the multi-drop bus and is used to provide fan-out for the I/O bus. A switch may provide peer-to-peer communication between different endpoints and this traffic, if it does not involve cache-coherent memory transfers, need not be forwarded to the host bridge. The switch is shown as a separate logical element but it could be integrated into a host bridge component. The low signal-count, point-to-point connections may be constructed with connectors and cables. The PCI Express mechanicals will enable new classes of system.

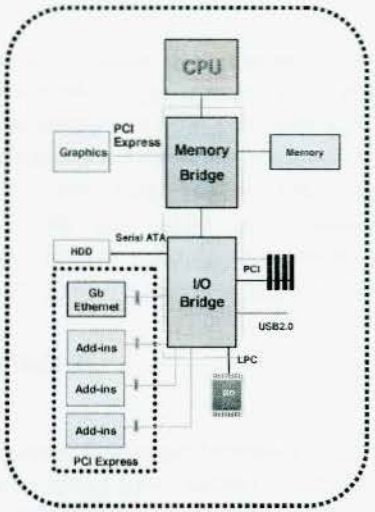


Figure 9: PCI Express

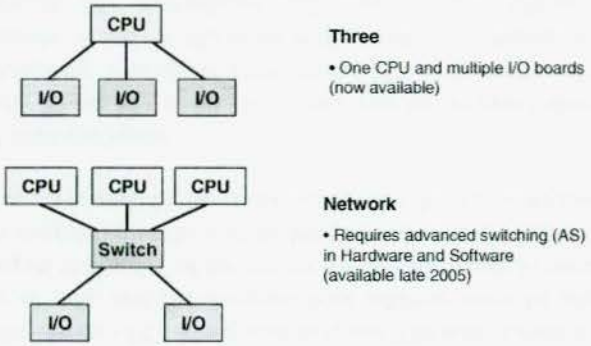


Figure 10: PCI Express architectures

The PCI Express Architecture is specified in layers as shown in Figure 11. Compatibility with the PCI addressing model load-store architecture with a flat address space) is maintained to ensure that all existing applications and drivers operate unchanged. PCI Express configuration uses standard mechanisms as defined in the PCI Plug-and-Play specification.

The software layers will generate read and write requests that are transported by the transaction layer to the I/O devices using a packet-based,

split-transaction protocol. The link layer adds sequence numbers and CRC to these packets to create a highly reliable data transfer mechanism. The basic physical layer consists of a dual-simplex channel that is implemented as a transmit pair and a receive pair. The initial speed of 2.5 Gigatransfers per second per direction provides a 200 MByte/s communications channel that is close to twice the classic PCI data rate.

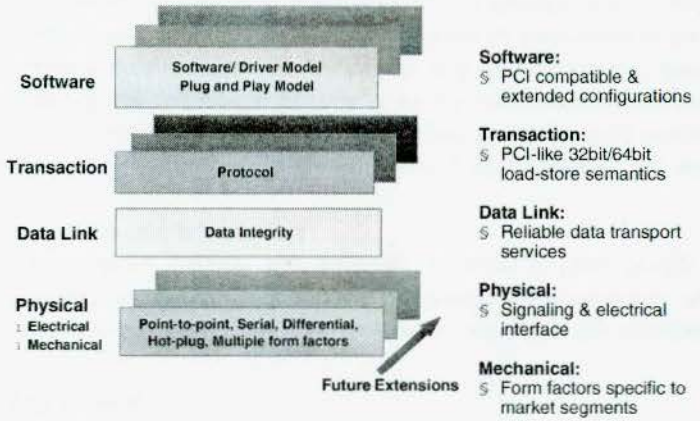


Figure 11: Layered PCI architecture enables modularity, reuse, and scalability

The fundamental PCI Express link consists of two, low-voltage, differentially driven pairs of signals: a transmit pair and a receive pair as shown in Figure 8. A data clock is embedded using the 8B/10B encoding scheme to achieve very high data rates. The initial frequency is 2.5 Gigatransfers per second per direction and this is possible to increase with silicon technology advances of up to 10 Gtransfers per second per direction (the theoretical maximum for signals in copper). The physical layer transports packets between the link layers of two PCI Express agents.

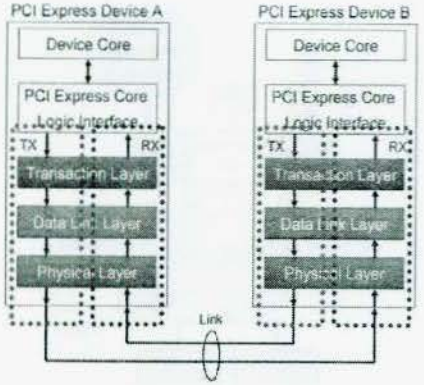
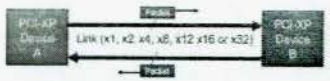


Figure 12: PCI Express device layers

The bandwidth of a PCI Express link may be linearly scaled by adding signal pairs to form multiple lanes. The physical layer supports x1, x2, x4, x8, x12, x16 and x32 lane widths and splits the byte data as shown in Figure 9. Each byte is transmitted, with 8b/10b encoding, across the lane(s). This data disassembly and re-assembly is transparent to other layers.

Point-to-Point Protocol

x1, x2, x4, x8, x12, x16 or x32 point-to-point Link



Differential Signaling

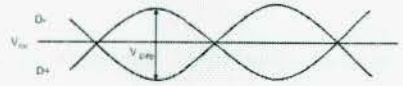


Figure 13: PCI Express bus

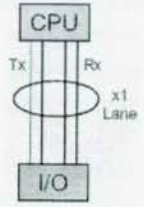


Figure 14: PCI Express lanes

The primary role of a link layer is to ensure reliable delivery of the packet across the PCI Express link. The link layer is responsible for data integrity and adds a sequence number and a CRC to the transaction layer packet. Most packets are initiated at the Transaction Layer. A credit-based, flow control protocol ensures that packets are only transmitted when it is known that a buffer is available to receive this packet at the other end. This eliminates any packet retries, and their associated waste of bus bandwidth due to resource constraints. The Link Layer will automatically retry a packet that was signaled as corrupted.

Differential Transmitter and Receiver

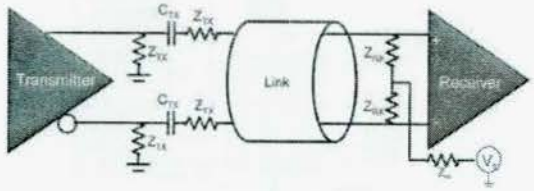
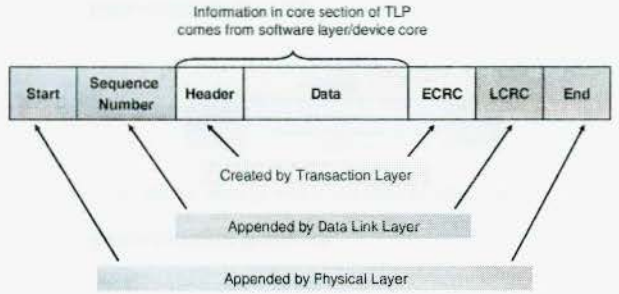


Figure 15: Physical layer of PCI Express

The transaction layer receives read and write requests from the software layer and creates request packets for transmission to the link layer. All requests are implemented as split transactions and some of the request packets will need a response packet. The transaction layer also receives response packets from the link layer and matches these with the original software requests. Each packet has a unique identifier that enables response packets to be directed to the correct originator. The packet format supports 32-bit memory addressing and extended 64-bit memory addressing. Packets

also have attributes such as "no-snoop", "relaxed-ordering" and "priority" which may be used to optimally route these packets through the I/O subsystem.



ECRC: End-to-End Cyclic Redundancy Check
LCRC: Link Cyclic Redundancy Check
TLP: Transaction Layer Packet

Figure 16: Transaction layer packets

Figures below show typical platforms using the PCI Express Architecture.

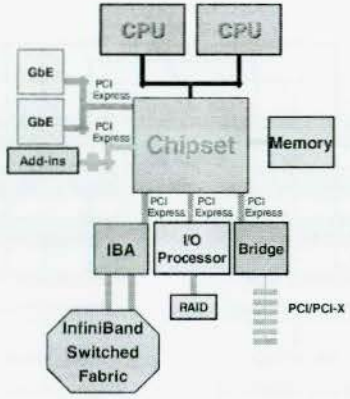


Figure 17: PCI Express in server platforms

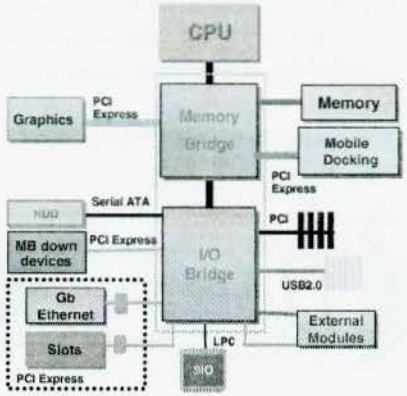


Figure 18: PCI Express in client platforms

The multiple, similar parallel buses of today's platform are replaced with PCI Express links with one or more lanes. Each link is individually scalable by adding more lanes so that additional bandwidth may be applied to those links where it is required – such as graphics in the desktop platform and bus bridges (e.g. PCI Express-to-PCI-X) in the server platform.

A PCI Express switch provides fan-out capability and enables a series of connectors for add-in, high performance I/O. The switch is a logical element that may be implemented within a component that also contains a host bridge, or it may be implemented as a separate component. It is expected that PCI will coexist in many platforms to support today's lower bandwidth applications until a compelling need, such as a new form factor, causes a full migration to a fully PCI Express based platform.

The server platform requires more I/O performance and connectivity including high bandwidth PCI Express links to PCI-X slots, Gigabit Ethernet and an InfiniBand fabric. The combination of PCI Express for "inside the box" I/O, and InfiniBand fabrics for "outside the box" I/O and cluster interconnect allows servers to transition from "parallel shared buses" to high speed serial interconnects.

The networking communications platform could use multiple switches for increased connectivity and Quality of Service (QOS) for differentiation of

different traffic types. It too would benefit from a multiple PCI Express links that could be constructed as a modular I/O system.

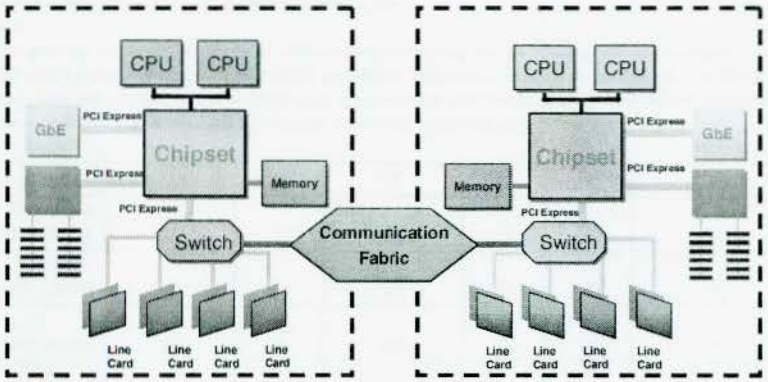
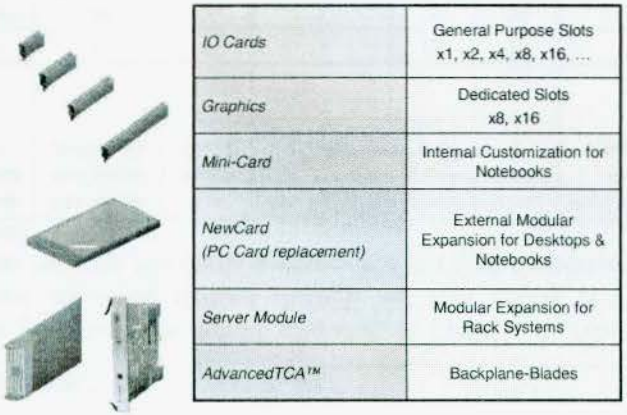


Figure 19: PCI Express in communication platforms

Figure 20 shows some of the form factors that can be used in a PCI Express system. Initial implementations of PCI Express-based add-in cards will co-exist alongside the current PCI-form factor boards (both full-size and "half-height").



IO Cards	General Purpose Slots x1, x2, x4, x8, x16, ...
Graphics	Dedicated Slots x8, x16
Mini-Card	Internal Customization for Notebooks
NewCard (PC Card replacement)	External Modular Expansion for Desktops & Notebooks
Server Module	Modular Expansion for Rack Systems
AdvancedTCA™	Backplane-Blades

Figure 20: PCI Express form factors

As an example connection; a higher bandwidth link, such as a sixteen-lane connection to a graphics card, will use a new connector placed alongside the existing PCI or AGP connector in the area previously occupied by those type of connectors. In addition to the base implementations of PCI Express-based add-in cards that replace AGP and PCI cards, other form factors are in progress of being developed.

1.3 Small Computer System Interface (SCSI)

SCSI is a family of disk interface systems that provide access services for peripheral I/O devices such as disk drives, CD-ROM discs, optical disks, tape drives, scanners, and other devices. Theoretically, you can plug any vendor's SCSI device into any SCSI controller. Each I/O device is called an LU (Logical Unit) and each LU is assigned an LUN (Logical Unit Number).

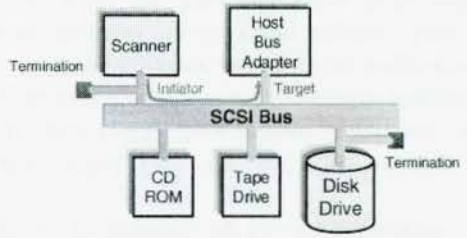


Figure 21: Small computer system interface (SCSI) basics

SCSI is normally implemented in host adapters, which provides a shared bus to which peripheral de-vices attach. Applications make service requests to SCSI, and SCSI issues commands to the logical units that execute the commands. All commands and data cross the SCSI shared bus, which is a parallel interface. Some of the features of the SCSI interface are described here:

- The original SCSI standard supports up to 7 devices on a single host adapter, but new standards support high-speed operation with up to 16 devices and bus lengths of up to 25 meters.
- SCSI devices are "smart" devices with their own control circuitry. They can "disconnect" themselves from the host adapter to process tasks on their own, thus freeing up the bus for other transmissions.
- The bus can handle simultaneous reads and writes.

SCSI has gone through several upgrades, as outlined in Table 1. New generations are being defined through the joint efforts of the ANSI (American National Standards Institute) X3T10 (SCSI) Committee and the SCSI Trade Association.

SCSI Trade Association Terms	Maximum Bus Speed [MByte/s]	Bus Width [bits]	Maximum Bus Length [meters] *LVD: [low-voltage differential] signaling HVD: [high-voltage differential] signaling*			Maximum No. of Devices Supported
			Single Ended	LVD	HVD	
SCSI-1[1]	5	8	6	[2]	25	8
Fast SCSI[1]	10	8	3	[2]	25	8
Fast/Wide SCSI	20	16	3	[2]	25	16
Ultra SCSI[1]	20	8	1.5	[2]	25	8
Ultra SCSI[1]	20	8	3			4
Wide Ultra SCSI	40	16		[2]	25	16
Wide Ultra SCSI	40	16	1.5			8
Wide Ultra SCSI	40	16	3			4
Ultra2 SCSI[1, 3]	40	8	[3]	12	25	8
Wide Ultra2 SCSI[3]	80	16	[3]	12	25	16
Ultra2 SCSI or Ultra160[5]	160	16	[3]	12	[4]	16
Ultra320	320	16	[3]	12	[4]	16

[1] Use of the word "Narrow" preceding SCSI, Ultra SCSI, or Ultra2 SCSI is optional.
[2] LVD was not defined in the original SCSI standards for this speed. If all devices on the bus support LVD, then 12-meter operation is possible at this speed. However, if any device on the bus is single-ended only, then the entire bus switches to single-ended mode and the distances in the single-ended column apply.
[3] Single ended is not defined for speeds beyond Ultra.
[4] HVD (Differential) is not defined for speeds beyond Ultra2.
[5] After Ultra2 all new speeds are wide only.

Table 1: SCSI trade association-endorsed terminology for SCSI parallel interface technology

The original SCSI was standardized in the late 1980s. It is characterized by a 50-pin connector. By the 1990s, enhancements were made, such as changing the physical connector to a 68-pin connector. New bus widths (more data

lines) were devised. These enhancements are called Wide SCSI. They include bus widths of 16 bits or 32 bits, allowing higher data transfer rates and addressing of up to 16 devices instead of the original 8. In addition, fast transfer rates are achieved by using synchronous data transfers instead of asynchronous data transfers.

Ultra SCSI (also called Fast 20) was the next advancement. It uses new SCSI chip sets with internal clock speeds that are doubled, thus doubling the megabyte-per-second transfer rates. Basically, Ultra SCSI doubles the transfer rate independent of the bus width. Thus, the original 8-bit SCSI-1 is boosted to 10 MByte/s by applying the Fast enhancements.

Ultra2 SCSI (also called Fast 40) was one of the most significant advancements. It uses LVDS (low-voltage differential signaling), which improves the signal-to-noise ratio and allows cable lengths up to 12 meters. Up to this point, SCSI implementations used the single-ended bus, which is a signaling scheme that has limited cable lengths. These lengths must be strictly adhered to since the Fast and Ultra SCSI enhancements were achieved by doubling the clock rate, and thus the fundamental frequency at which data is transferred. Doing so required halving the cable lengths to prevent signal degradation.

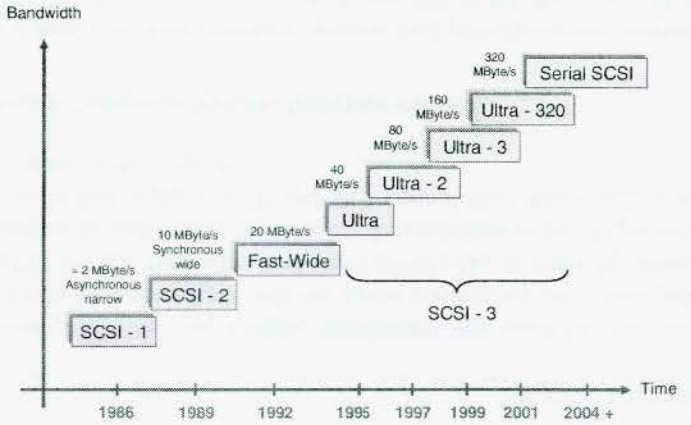


Figure 22: SCSI History

The LVDS signaling scheme used with Ultra2 SCSI improves performance, but the differential bus pushes down the number of devices supported on the bus. LVDS uses extremely low voltage levels, which translates to low radiation and less power consumption. Noise is reduced by sending signals across two wires with opposing voltage levels. The receiver reads the difference in voltage levels and rejects the noise. LVDS is an important signaling technology used in other interconnection schemes such as InfiniBand.

The current generation, Ultra3 SCSI, was defined in the late 1990s. It defines several new generations of SCSI, including Ultra160 SCSI (160 MByte/s), Ultra320 SCSI (320 MByte/s), and Ultra640 SCSI (640 MByte/s). Ultra 160 SCSI products started shipping in 1999 and Ultra320 SCSI products were appearing in 2001. Ultra3 SCSI adds features such as the ability to choose the highest possible data transfer rate; packetization (transfer multiple commands and messages at once); and QAS (quick arbitrate and select), which provides faster arbitration to reduce connect/disconnect on the bus. Other recent SCSI developments include VHDCI (Very High Density Cable Interconnect), LUN bridging, and SCSI switching. VHDCI provides high-speed interconnections where space is limited. With LUN bridges, 960 devices can be connected, while expanders allow SCSI cables to extend up to 75 meters. SCSI switching provides the same benefits as other switching technologies.

New SCSI drives that rotate at 15,000 rpms have the potential to deliver a sustained data rate of over 40 MByte/s. A typical server will have four drives, and those drives can produce a combined data rate of 160 MByte/s or more. Clearly, the new Ultra3 SCSI standards are needed. But these new systems must be installed in 64-bit PCI systems since the older 32-bit, 33-MHz PCI bus has a maximum data rate of 133 MByte/s.

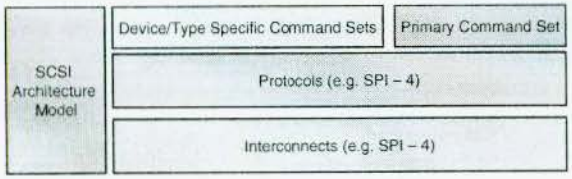


Figure 23: SCSI-3

SCSI-3 defines a number of different standards, each covering different aspects of SCSI. Therefore it is necessary to organize these into a format that defines how they relate to each other, and the goals of the interface as a whole. This structure is called the architecture of SCSI-3. SCSI-3 architecture is defined by a document called the SCSI-3 Architecture Model (SAM), which has been approved as ANSI standard X3.270-1996.

The SCSI-3 Architecture Model has several functions. An important one is to organize and categorize the various other standards that fall under SCSI-3. This serves to structure these standards in a way that makes sense to SCSI standards developers, hardware designers and users. The structure defines broad, generic requirements at a high level, which are refined to more specific low-level requirements through the use of particular implementation standards. Most of the different SCSI-3 documents fall into the following three general categories:

- Commands: These are standards that define specific command sets for either all SCSI devices, or for particular types of SCSI devices.
- Protocols: These standards formalize the rules by which various devices communicate and share information, allowing different devices to work together. These standards are sometimes said to describe the transport layer of the interface.
- Interconnects: These are standards that define specific interface details, such as electrical signaling methods and transfer modes. They are sometimes called physical layer standards as well.

SCSI-FCP (Small Computer System Interface-Fibre Channel Protocol) is an implementation of Fibre Channel that transports SCSI protocols. SCSI is a disk interface technology that normally runs over a parallel connection. SCSI-FCP is a serial SCSI that allows SCSI-based applications to use an under-lying Fibre Channel connection. SCSI-FCP is widely used to connect high-performance servers to storage subsystems, especially in the SAN environment. It provides higher performance (100 Mbit/s), supports cable lengths up to 10 km, and can address up to 16 million devices. Data is transferred in frames rather than blocks.

Serial attached SCSI (SAS) takes advantage of existing SCSI development, in that it makes full use of the SCSI command set that has been developed and

proven over the years. Because of its serial interconnect; it is also a major departure from the older physical technology. The application layer and command set remain the same, however, so users can make the transition easily. The move from parallel to serial connections became necessary when SCSI development hit performance obstacles upon achieving transfer rates above 320 megabytes/second (Ultra320 SCSI).

Interconnect	Standard	Typical Speed
Fibre Channel	FCP	100 MByte/s
Serial Storage Architecture (SSA)	SSA-S2P, SSA-TL1, SSA-PL1	20 MByte/s
Serial Storage Architecture (SSA)	SSA-S3P, SSA-TL2, SSA-PL2	40 MByte/s
FireWire (IEEE 1394)	SBP-2	50 MByte/s
Fibre Channel	FCP-2	200 MByte/s
InfiniBand	SRP	250 MByte/s
Gigabit Ethernet	iSCSI	100 MByte/s

Table 2: Serial SCSI supports different transport technologies

The requirements for implementing parallel data transfer, with all signals departing and arriving in parallel over longer distances and at greater speeds, became prohibitive. Use of a serial bus eliminates this timing requirement for signals, and enables much additional efficiency.

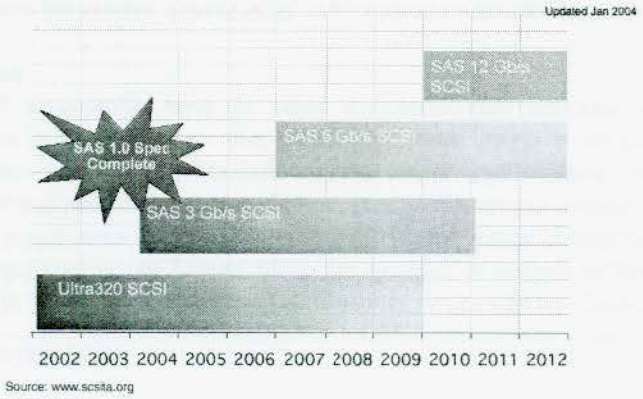


Figure 24: Serial atteched SCSI (SAS) roadmap

1.3.1 Parallel SCSI

The SCSI bus connects all parts of a computer system so that they can communicate with each other. The bus frees the host processor from the responsibility of I/O internal tasks.
The SCSI protocol is a peer-to-peer relationship: one device does not have to be subordinated to another device in order to perform I/0 activities. A total of eight devices can be connected to the bus simultaneously. Only two of these devices can communicate on the bus at any given time.

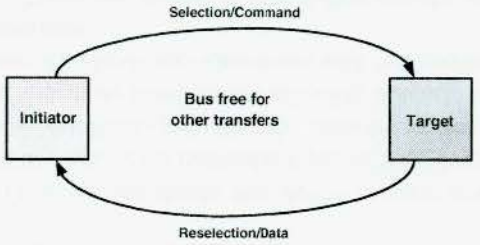


Figure 25: Operation of the SCSI bus

The Small Computer System Interface bus can be time-shared, which results in greater usage of bus bandwidth. This is how it works: while one device is using the bus, other devices may be active and performing internal activities. Devices do not use the bus unless they are involved in data transfer or have status to report. Devices may disconnect from the bus while time-consuming activities internal to the device are occurring. As soon as a device is ready to resume communication, the device can arbitrate for the bus (when the bus is free) to reattach to the host. System performance is significantly increased when devices disconnect and reconnect to the bus. During the bus phases (Figure 26), devices must first contend for access to the bus. Then a physical path is established between the initiator and target. Remember, the SCSI bus cannot be in more than one phase at a time.

On a busy system, the SCSI bus may be free for a short time while there is no device requesting the bus, or it could remain in the bus free state indefinitely.

A device can arbitrate and win the bus in 3.6 ps or less. Devices that lose arbitration can try again when the bus is free. The Selection phase can occur in 580 ns. If the target does not respond, the bus is free in about 250 ms. The target controls the bus alter the selection phase. This is stile first information transfer phase in the connection. It allows the initiator to send an Identify message to the target. Messages are always transferred asynchronously. Six Inquiry command bytes are transferred asynchronously from the initiator to the target. The target responds with Inquiry data. The data is transferred synchronously it both the target and the initiator have previously established a synchronous data transfer agreement. The target sends a single status byte asynchronously. The last information that is transferred in the connection is typically the Command Complete message. The target releases all signals within 10ls of the initiator's acknowledgement of the previous message. The initiator releases any signals it may be driving within 800 ns of its detection of the BSY signal.

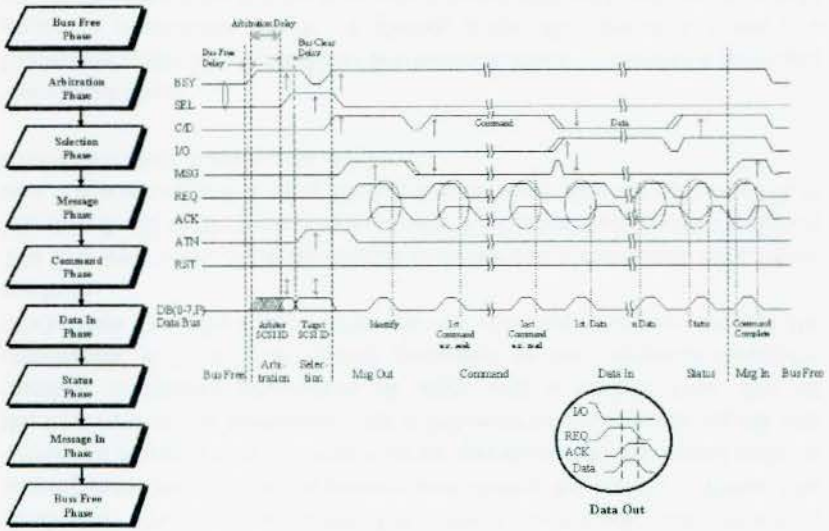


Figure 26: SCSI bus phases

The SCSI bus uses eighteen signals. Nine are control signals used to develop logical bus phases, and nine are data signals, including parity, for messages, commands, status, and data. The state of the SEL, BSY, and I/O signals and

the sequence of the phases determine when the Bus Free, Arbitration, Selection, and Reselection phases are entered.

A description of SCSI signals is as follows:

- BD(0)#-BD(7)#: data bits
- BD(P)#: parity bit
- BSY# (busy): the signal indicates whether the bus is currently busy
- SEL# (select): the signal is used by the initiator to select the target device; on the contrary, the target may also use SEL to re-establish the connection to the initiator after a temporary release of the bus control
- C/D# (control/data): the signal is exclusively controlled by the target, and indicates whether control information or data is present on the SCSI bus. An active signal (with a low level) denotes control information
- ATN# (attention): an initiator activates the signal to indicate the attention condition
- RST (reset): an active signal resets all connected SCSI devices
- I/O# (input/output): the signal is exclusively controlled by the target device, and indicates the direction of the data flow on the data bus relative to the initiator. An active signal (with a low level) means a data transfer to the initiator
- MSG# (message): the signal is activated by the target during the message phase of the SCSI bus
- REQ# (request): the signal is activated by the target unit to indicate the handshake request during the course of a REQ/ACK data transfer
- ACK# (acknowledge): the signal is activated by the initiator to indicate the handshake acknowledge during the course of a REQ/ACK data transfer

The Selection or Reselection phase can be entered only from the Arbitration phase. The Arbitration phase can be entered only from the Bus Free phase. The Bus Free phase can be entered from any of the other phases (although some transitions are caused by errors). To determine the current phase, you need to know information about the previous phase and the state of the signals. The initiator and target drive these signals to change from one phase to another phase.

A request to a peripheral device is performed by sending a command descriptor block to the target. For several commands, the request is accompanied by a list of parameters sent during the Data Out Phase. The command descriptor block always has an operation code (command code) as the first byte of the command. This is followed by an optional logical unit number, command parameters (if any), and a control byte. For all commands, if there is an invalid parameter in the command descriptor block, then the target shall terminate the command without altering the medium.
The first byte of all SCSI commands shall contain a command code. Three bits (bits 7 - 5) of the second byte of each SCSI command specify the logical unit if it is not specified using the Identify message. The last byte of all SCSI commands shall contain a control block.

Command Code:
Command codes are divided into two sections. Bits 5 - 7 identify a particular group of commands. Bits 0 - 4 identify a specific command. There is a possibility, therefore, to have 8 groups of commands and each group to hold 32 commands. Most of the commands are contained within group 0.

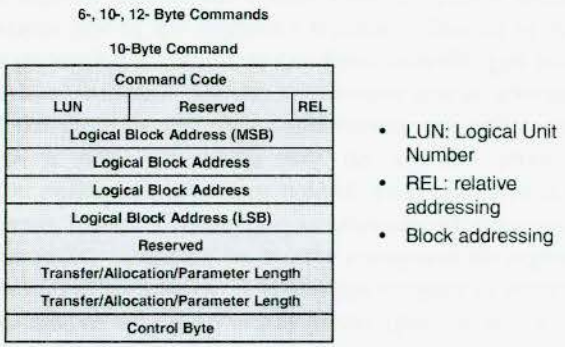


Figure 27: SCSI commands

The group code specifies one of the following groups:

- Group 0 - six-byte commands
- Group 1 - ten-byte commands
- Group 2 – twelve- byte Commands

- Group 3 - reserved
- Group 4 - reserved
- Group 5 - twelve-byte commands
- Group 6 - Digital Unique
- Group 7 - Digital Unique

Logical Unit Number:
Digital drives support only one logical unit, LUN=0. The target shall ignore the logical unit number specified within the Command Descriptor Block if an Identify message was received.
Logical Block Address:
The logical block address begins with block zero and is contiguous up to the last logical block on the drive. A six-byte command descriptor block contains a 21-bit logical block address. The ten-byte and the twelve-byte command descriptor blocks contain 32-bit logical block addresses. Logical block addresses in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

Transfer Length:
The transfer length field specifies the amount of data to be transferred, usually the number of logical blocks. For several commands the transfer length indicates the requested number of bytes to be sent as defined in the command description. For these commands the transfer length field may be identified by a different name. See the following descriptions and the individual command descriptions for further information. Commands that use one byte for transfer length allow up to 256 blocks of data to be transferred by one command. A transfer length value of 1 to 255 indicates the number of blocks that shall be transferred. A value of zero indicates 256 blocks. Commands that use multiple bytes for transfer length allow 65,535 or greater blocks of data to be transferred by one command. In this case, a transfer length of zero indicates that no data transfer shall take place. A value of one or greater indicates the number of blocks that shall be transferred. Refer to the specific command description for further information.

Parameter List Length:
The parameter list length is used to specify the number of bytes sent during the Data Out Phase. This field is typically used in command descriptor

blocks for parameters that are sent to a target that is mode parameters, diagnostic parameters, log parameters.

Allocation Length:
The allocation length field specifies the maximum number of bytes that an initiator has allocated for returned data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The target shall terminate the Data In Phase when allocation length bytes have been transferred or when all available data have been transferred to the initiator, whichever is less. The allocation length is typically used to return sense data, for example mode data, log data, diagnostic data, to an initiator.

Control Byte:
The control byte is the last byte of every command descriptor block. The Flag, Link, and Digital Unique fields are not supported and must be set to zero. If any of the link, flag, or Digital unique bits are set to one, the drive shall return Check Condition status with the sense key set to Illegal Request.

1.3.2 Serial SCSI (SAS)

Serial Attached SCSI is the logical technical evolution that satisfies the future requirements of the enterprise data canter. It integrates two established technologies - SCSI and serial, combining the proven utility and reliability of the SCSI protocol with the performance advantages of serial architecture. The result is scalable I/O, which can move data at the speed of three gigabits per second per path, across multiple, low-cost full duplex paths.

Serial Attached SCSI (SAS) is a point-to-point architecture in which all storage devices connect directly to a SAS port rather than sharing a common bus as traditional SCSI devices do. Point-to-point links increase data throughput and improve the ability to locate and fix disk failures. More importantly, the SAS architecture solves the clock skew and signal degradation problems of parallel SCSI at higher signaling rates. SAS inherits its command set from parallel SCSI, frame formats from Fibre Channel, and physical characteristics from Serial ATA.

SAS is defined for full-duplex operations. This means that transfers may occur in both directions on a pathway simultaneously. The 3.0 Gbit/s transfer rate was deemed acceptable for the first generation of SAS devices, as it equates to a data transfer rate of 300 MByte/s per direction, which doubles to a maximum of 600 MByte/s per pathway. Also, several pathways per device may be aggregated to multiply bandwidth. The next transfer rate target for SAS is 6.0 Gbit/s per pathway with complete backward compatibility with first-generation SAS devices.

Point-to-point configurations provide for high bandwidth, but require intermediary devices between initiator devices (or hosts) and target devices (or peripheral devices) to provide a topology where there may be more than two devices in a system. Inexpensive expanders are the intermediary devices defined for SAS. SAS expanders allow for systems in which more than one initiator may have a connection to more than one target (as is allowed by parallel SCSI).

Another of the original goals for the SAS effort was to maintain or expand the maximum number of devices beyond the maximum number (16) that could exist in a parallel SCSI domain. A SAS system with expanders may address up to 16,256 devices in a single SAS domain. SAS expanders are also defined so that end devices connecting to expanders may be either SAS devices or SATA (Serial ATA) devices, satisfying the goal of allowing heterogeneous system configurations. Figure 28 shows an example of a maximum SAS system configuration.

The box in the center of the figure represents a fan-out expander. There may be one fan-out expander per SAS domain (effectively, a SAS system). A fan-out expander may connect to a maximum of 128 SAS devices. These may be any combination of edge expanders, initiators, or storage devices. The edge expanders can be connected to the fan-out expander. An edge expander may connect to a maximum of 128 SAS devices. This includes no more that one other expander, with the rest of the connected devices being either initiators or storage devices.

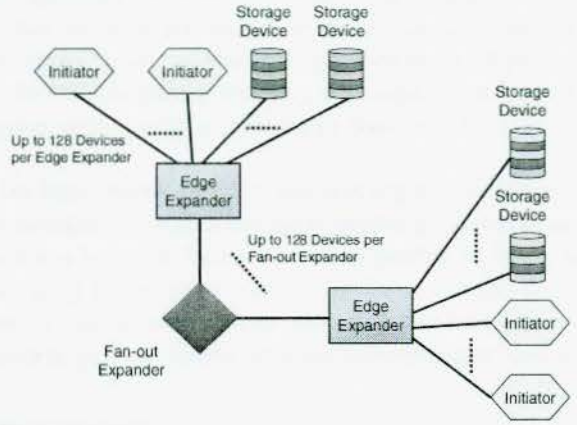


Figure 28: A maximum SAS system configuration

The SAS protocol highly leverages protocols described in other SCSI standards (such as the SCSI Parallel Interface, or SPI, standard). Transfer of commands, data, status, and other information for SAS devices is accomplished using packets very similar to the packets used for information unit transfers for parallel SCSI (as defined in SPI) and the packets defined for Fibre Channel. The format of SAS packets (called "frames") is almost identical to Fibre Channel packets. The payload of the packets consists of command descriptor blocks (CDBs) and other SCSI constructs defined in other SCSI standards (such as the SCSI Primary Command Set and SCSI Block Commands). Using SCSI protocol and architecture, it is possible to bridge from a SAS system to other systems using Infiniband, iSCSI, or Fibre Channel, which also use the same SCSI objects. Leveraging elements of the SCSI protocol, including Fibre Channel where appropriate, was another of the initial goals for SAS, as there was a strong desire on the part of all of the members of the working group to provide maximum compatibility with current infrastructure and minimize the risk and cost of transition to the new interface. The SAS protocol is divided into four layers: the phy layer, the link layer, the port layer and the transport layer. These four layers are contained in the SAS port. This means that applications (like software and drivers) used to communicate with parallel SCSI ports may also be used to communicate with SAS ports with little or no modification. Figure 29 shows the relationships between the protocol layers in a SAS device.

The application layers include application and driver software. The application layers create specific tasks for the transport layer to process. The transport layer encapsulates commands, data, status, etc., into SAS frames and assigns these to the port layer. The transport layer also receives SAS frames from the port layer, disassembles the frames, and sends the frames' content to the application layer. The SAS port layer receives the packets from the transport layer and sends messages to the link layer in order to establish connections. The port layer is also responsible for selecting phys on which to transmit frames in multi-pathway devices. Once connections are established, the port layer forwards the packets for transmission and receives packets that are then sent on to the transport layer. The SAS link layer controls the phy layer for connection management. The SAS phy layer contains the hardware (such as transceivers and the encoding machines) that connects to the SAS physical interface and sends the signals out on the wire.

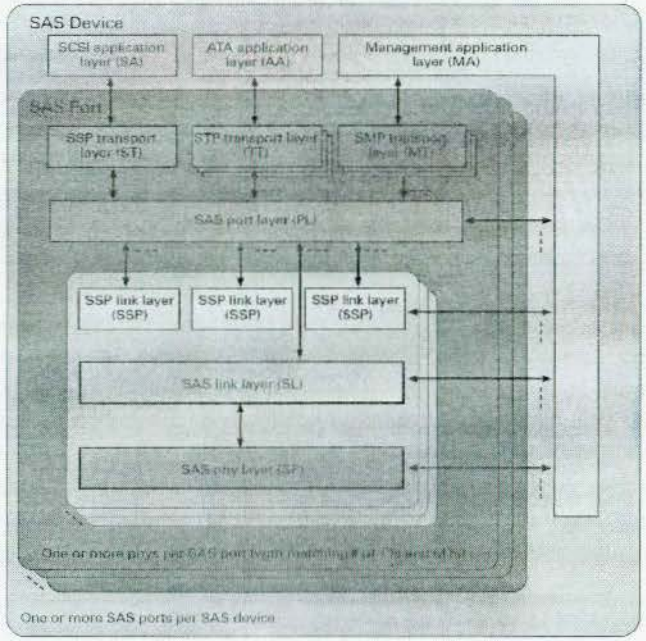


Figure 29: Protocol layers in a SAS device

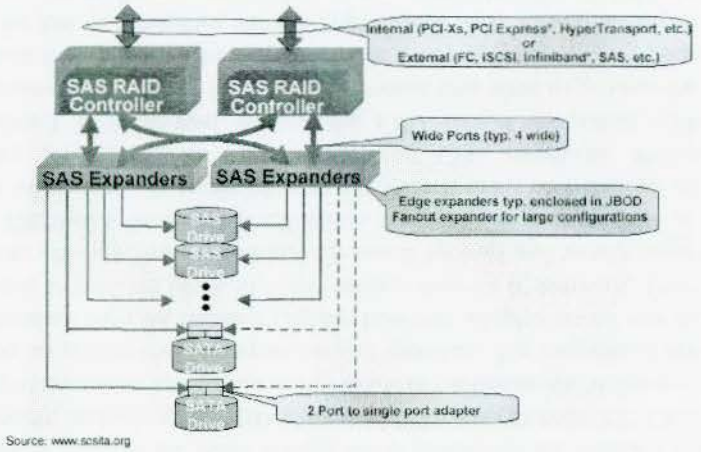


Figure 30: Typical serial SCSI (SAS) configuration

Typical applications of Serial Attached SCSI are shown in Figures Figure 30 Figure 31. These applications include enterprise storage environments, high-end desktop and workstations, internal servers, SAN (storage arera network), NAS (networked attached storage), DAS (direct attach storage) with external enclosures, rack-mount servers with JBOD storage, and disk-based backup storage.

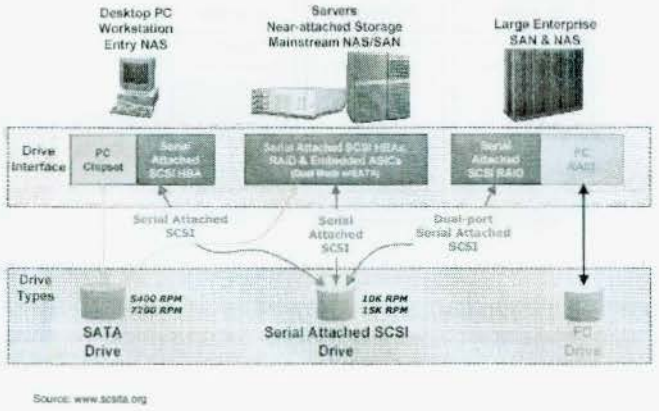


Figure 31: Serial attached SCSI (SAS) applications

1.3.3 Internet SCSI (iSCSI)

Internet SCSI (iSCSI) is an emerging standard that defines the encapsulation of SCSI packets in TCP and routed using IP. Its development allows block-level storage data to be transported over widely used IP networks, enabling data access from anywhere, effectively eliminating the physical boundaries of the storage network. iSCSI enables block level data to be accessed over a standard Ethernet/IP network whether it resides on a direct attached SCSI-based device or a Fibre Channel SAN.

Figure 32: Internet SCSI (iSCSI)

With iSCSI, enterprises and storage service providers (SSP) can build global storage networks and manage them from a central location using existing IP network infrastructures. iSCSI enables standard SCSI commands to be passed between host systems (initiators) and storage devices (targets) over an Ethernet/IP network.

- iSCSI Standards

The iSCSI standard has been prepared by SNIA and ratified by the Internet Engineering Task Force (IETF) in 2003. Numerous systems, storage, and networking industry leaders are developing and testing iSCSI solutions to meet the immediate need for this functionality and will introduce interoperable solutions that will support the evolving iSCSI standard.

- Classes of iSCSI Host Adapters

iSCSI network interface cards resolve many of the outstanding issues associated with Fibre Channel Host Bus Adapters (HBAs) and fabrics by bringing block storage I/O into traditional IP networking. By eliminating unique Fibre Channel requirements at the host end and the network, server administrators can leverage well-understood Gigabit Ethernet technology and IP networking to reduce ongoing support and management costs.

In addition, the customer has several options for iSCSI implementation, depending on price sensitivity, bandwidth requirements, and host CPU overhead requirements. As shown in Figure 33 these adapter options

include software-only solutions, embedded TCP off-load engines, and combined TCP/IP off-load and iSCSI in silicon.

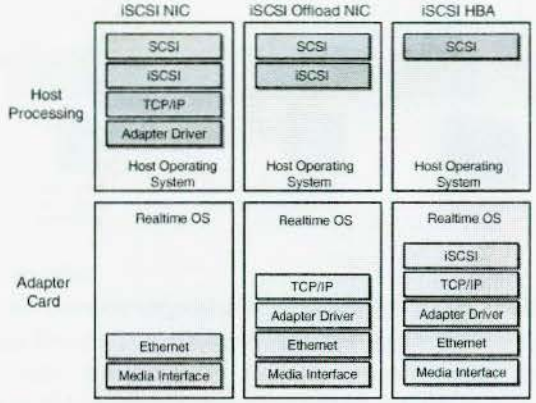


Figure 33: iSCSI adapter implementation options

iSCSI in Software

The iSCSI protocol for block storage I/O is predicated on standard Ethernet transports, including Fast (100 Mbit/s) and Gigabit Ethernet (1.25 Gbit/s). As a very economical, low-performance block I/O solution, iSCSI drivers can be run on an off-the-shelf Ethernet card or interface. For example, a server or workstation with an existing Gigabit Ethernet adapter could run iSCSI in software to perform block I/O tape backup over an IP network. This would be more efficient than traditional file-based backup and would require less than 20MByte/s bandwidth utilization. Software implementation of iSCSI offers advantages in terms of cost; however it must be balanced with the penalty it imposes in performance and CPU overhead. Without an embedded TCP off-load engine, the CPU would be tasked with TCP processing. In addition, the host CPU would also have to process the iSCSI protocol that is carried by TCP/IP. While iSCSI in software may be suitable only for low performance storage applications such as tape backup, it does offer a new mid-range option for storage networking unavailable in Fibre Channel SANs. For gigahertz and higher speed server CPUs, the overhead of TCP and iSCSI processing may be an acceptable tradeoff for the convenience of iSCSI storage access.

iSCSI in Software with TCP Off-load

The development of TCP off-load engines (TOEs) marks a significant advance for all TCP/IP-based protocols, including iSCSI. While TCP provides session integrity in potentially congested or unstable networks, it incurs significant processing overhead on the host CPU. Off-loading this processing to a host network interface card frees host CPU cycles and enables much higher performance solutions.

Since block I/O transactions may engender sustained high volumes of TCP/IP exchanges, IP storage is a direct beneficiary of TOE technology. With the burden of TCP overhead removed from the host CPU, only iSCSI processing is required. The remaining challenge in terms of CPU utilization is to optimize iSCSI handling so that blocks of data can be served more efficiently to the host. With clever engineering, wire-speed performance can be achieved when running software iSCSI on an optimized TOE accelerated network adapter.

iSCSI in Silicon with TCP Off-load

In some vendor implementations, high performance iSCSI adapters off-load both TCP and iSCSI processing to the interface card. While this adds cost to the adapter, it provides both wire-speed iSCSI transport and minimal CPU overhead. The iSCSI processing may be performed in a custom ASIC or an on-board processor. In either case, the SCSI commands, status and data encapsulated by iSCSI are served up by the adapter card directly to the host operating system.

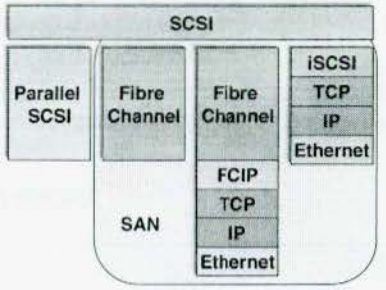


Figure 34: SCSI in storage area networks (SANs)

1.4 HyperTransport (HT)

HyperTransport (HT) technology is defined by channel topology, signal electrical characteristics, and command/address/data packet protocols. Topology defines the structure of the HyperTransport link. Electrical protocols define the physical characteristics of the HyperTransport signal interface. The packet protocols define how data is organized and transferred across the HyperTransport link.

HT defines direct connection to the CPU or several CPUs via high bandwidth, very low latency links. These links can be 16-bit or 8-bit HT links. With HyperTransport Specification 2.0, top clock speed is 1.4 GHz dual-data rate or DDR. Data throughput of 2.8 Gigatransfers/second per signal pair can be achieved, supporting up to 22.4 Gigabytes/second aggregate data throughput. HT enables in-system connectivity of high performance peripheral functions (e.g. security, storage, ...)

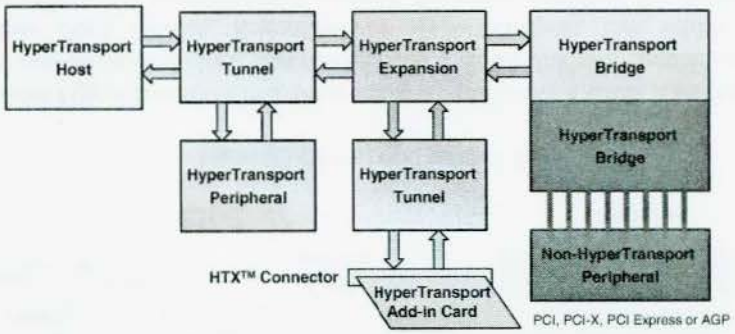


Figure 35: HyperTransport connections

The HyperTransport daisy-chain topology includes a required "host" device, at least one end-point or "cave," optional "tunnel" devices that connect the link to other HyperTransport devices and optional "bridge" devices that interface with non-HyperTransport interconnect technologies. There can be a maximum of 32 HyperTransport-enabled devices in a single daisy chain, although tunnels and bridges can be employed to create tree structures where additional, but separate HyperTransport daisy chains can

be linked to the first chain. HyperTransport switch devices, defined in Specification 1.05 can be deployed to create switched topology systems.

HyperTransport-enabled devices are configured to be one of four types. The host, often a CPU, is always considered the top of the link and traffic from the host is downstream while traffic to the host is upstream. A cave serves as an endpoint to a HyperTransport link. A bridge can be configured as an endpoint with a secondary interface to other interconnect technologies, or as a tunnel with an interface to other technologies.

The HyperTransport bridge device enables HyperTransport links to connect to other I/O technologies such as PCI, PCI-X, PCI Express or AGP devices.

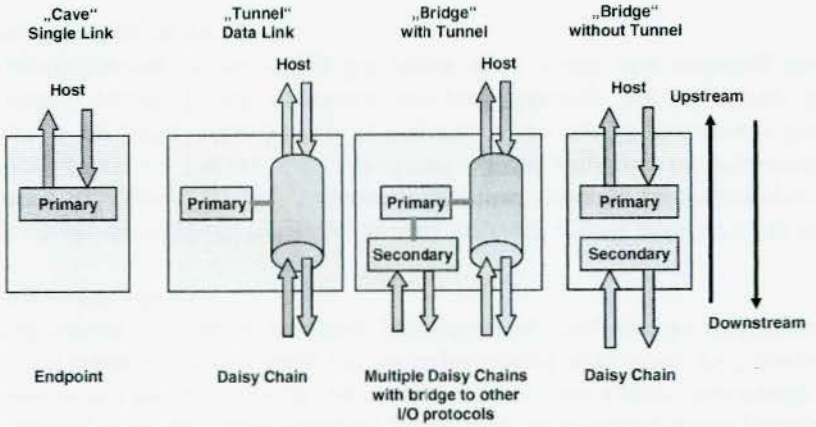


Figure 36: HyperTransport device configurations

The HyperTransport Link consists of a set of command/address/data (CAD) lines (ranging from 2, 4, 8, 16, or 32 bits wide), one control line per link, and one clock line per eight bits of CAD. Command and data information carried on the CAD lines is assembled into HyperTransport control and data packets. Control packets are 4 or 8 bytes long and 4- to 64-byte data packets follow read and write control packets. The state of the CTL line determines what type of packet, control or data, is being carried on the CAD lines.

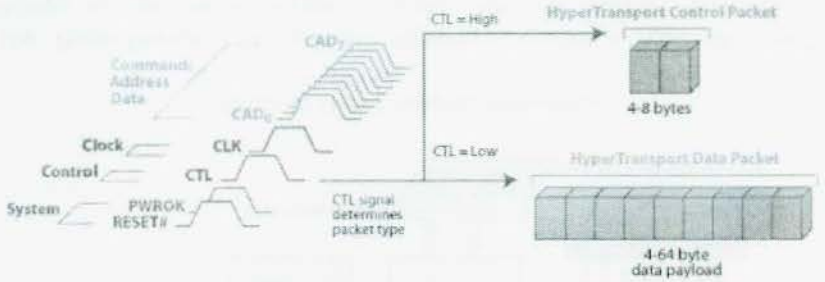


Figure 37: 8-bit HyperTransport link

A 16-bit HyperTransport link shows that an additional 8 CAD lines require only one additional clock signal. Likewise, a 32-bit link will require only 2 further clock signals. Regardless of HyperTransport link width, the HyperTransport packet format remains the same, enabling the design of asymmetrical links (one link wider than the other).

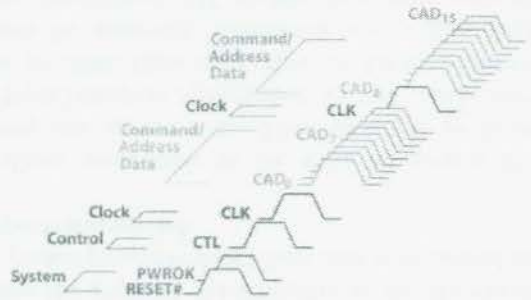


Figure 38: 16-bit HyperTransport link

The HyperTransport link uses balanced LVDS lines. Signal voltage on each wire line is symmetrical and of opposite polarity, ensuring the highest noise immunity. Each LVDS line is implemented by means of twin wire lines – otherwise called balanced, or differential line – carrying electrical signals that are equal in amplitude and timing but with opposite polarity. The secular nature of the signals carried over the balance line prevents electrical noise within the system from affecting and potentially corrupting the signal detection process at the receiver end - noise would affect both signals in

equal measure thereby nullifying the effect - thus ensuring a high degree of architectural noise immunity, as well as a maximized transmission range. Even though there are two wire lines per signal line, requiring a second printed circuit board (PCB) trace for each data pin/pair, the multiple advantages of higher operational speeds, higher noise immunity and lower power consumption more than make up for this minor disadvantage. In addition, since the HyperTransport protocol of packet-based traffic greatly reduces the number of total signal lines required for a given bandwidth, HyperTransport architectures can be implemented with fewer PCB traces and require less signal shielding precautions as compared to traditional single-ended buses.

A simple de-emphasis technique is used for clock speeds over 1.0 GHz in order to support the 2.8 Gigatransfers/second links of HyperTransport Specification 2.0. The eye of the signal uses reduced amplitude for sequential bits of the same value. High-pass filtering on the output counteracts the printed circuit board material's low-pass filtering roll-off above 2 Gigabits/second by extending the signal eye's width and reducing the number of signal transitions.

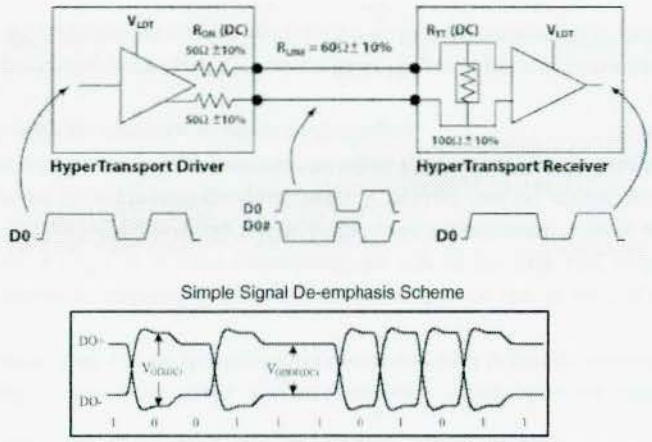


Figure 39: HyperTransport electrical specification

De-emphasis uses a 1 bit "history" to de-emphasize the differential amplitude generated by the transmitter when transmitting a continuous run of 1's or 0's. Since the de-emphasis creates a smaller, but cleaner eye signal at the receiver end, the receiver requires a higher sensitivity.

HyperTransport control packets consist typically of 4 to 8 bytes of command information. With optional 64-bit extended addressing, control packets can be 12 bytes. Data packets consist of 4- to 64-byte data payloads (in increments of 4 bytes) and directly follow either 1) an 8-byte read request followed by a 4-byte read response or 2) an 8-byte write request control packet.

Figure 40: HyperTransport packet format

Single control line is used to determine when the link is carrying a control packet (the control signal is asserted) or a data packet (the control signal is de-asserted).

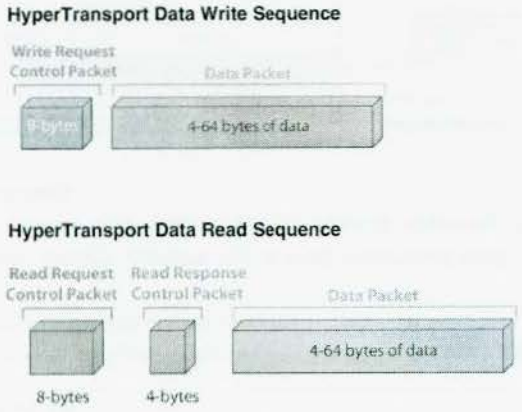


Figure 41: HyperTransport reads and writes

HyperTransport commands and data are separated into one of three types of virtual channels: non-posted requests, posted requests and responses. Non-posted requests require a response from the receiver. All read requests and some write requests are non-posted requests. Posted requests do not require a response from the receiver. Write requests are posted requests. Responses are replies to non-posted requests. Read responses or target done responses to non-posted writes are types of response messages.

HT reads and writes are very low overhead. Writes require only an 8-byte Write Request control packet followed by the data packet. Reads require an 8-byte Read Request control packet, followed by a 4-byte Read Response packet from the receiver, followed by the actual read data packet.

Priority request interleaving (PRI) enables a high priority request command (only 8-byte long) to be inserted within a potentially long, lower priority data transfer, so that higher priority traffic can be initiated concurrently. A typical use is shown in the figure below.

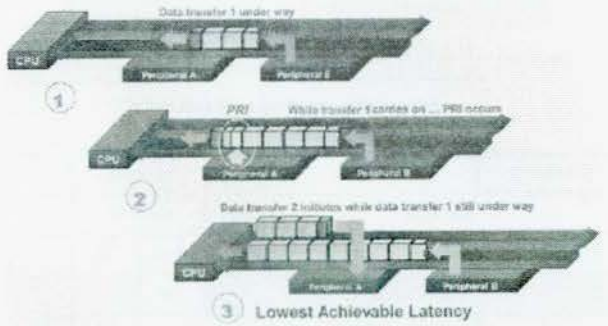


Figure 42: Priority request interleaving

While data transfer 1 is underway between peripheral B and the host, the need arises for peripheral A to start a data transfer from the host. Without PRI, transfer 2 would have to wait until transfer 1 completes and, should transfer 1 be the answer to a cache miss, for instance, latency for transfer 2 would become prohibitive. With PRI, a control packet is promptly inserted within transfer 1's data stream, instructing the link to initiate data transfer 2 on the other link channel concurrently with the completion of data transfer

1. This mechanism, unique to HyperTransport technology, greatly reduces latency of HyperTransport-based systems.

HyperTransport control packets use a 6-bit command field to indicate the packet type – write and read requests, special requests, responses, and information packets.

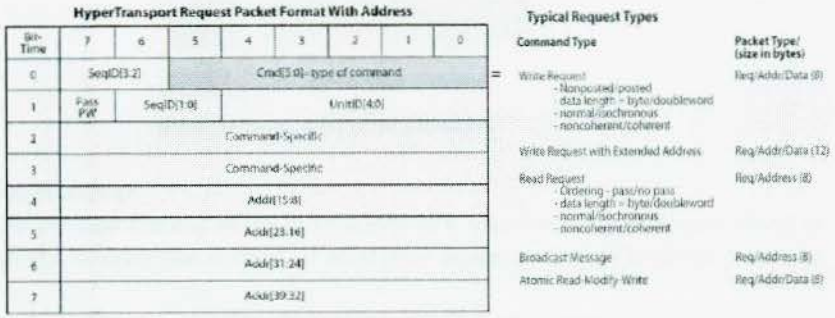


Figure 43: HyperTransport control packet

Base control packets are 4 or 8 bytes long (unless using extended 64-bit addressing, in which case they are 12 bytes long). They are divided into three general types: request packets, response packets and info packets. The Cmd[5:0] field determines the type of the control packet.

HyperTransport Read Response control packets showing the Command field and other important fields highlighted are shown in Figure 44. Read Response packets will be followed by a 4- to 64-byte data packet. The Byte Count is contained in the two fields, Count[1:0] and Count[3:2] and the Isoc bit is set depending on the type of bandwidth required for the data payload. Isochronous traffic such as multimedia data streams (information such as video frames or audio information that must arrive in sequence and without delay) are supported in HyperTransport using the Isoc bit. When set, it means that the accompanying data payload or data packet contains information that must be passed along with a higher priority than normal or non-isochronous traffic. The Error0 and Error1 bits give the status of the transaction. Normal completion is the rule, as the HyperTransport base link at speeds in excess of 1.0 GHz has been proven to an extremely reliable link.

Field tests have shown expected error rates to be 1 error per 100 years. This is on the order of SRAM interface error rates.

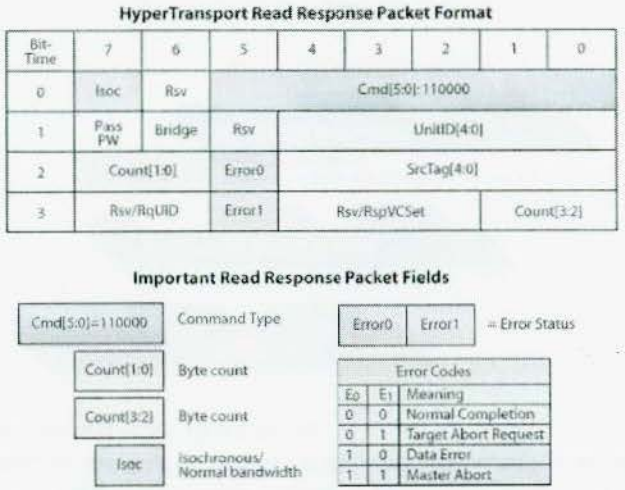


Figure 44: Read response packet format

Some of other common used command types are shown in Figure 45. By redefining some of the command field bits and accessing some reserved field bits, HyperTransport DirectPacket specification enables the carrying of user packet data in virtual channels within a standard-sized HyperTransport control packet (followed of course by a HyperTransport data packet).

HyperTransport DirectPacket carries user packet data in virtual channels specified by one or more virtual channel packets (redefined posted write control packets) and one or more data packets. If the user packet is equal to or less than 64 bytes, it can be placed in a single HyperTransport data packet. If it is more than 64 bytes, it can be placed in a sequence of data packets by breaking it up into 64-byte segments. If the last segment is less than 64 bytes, the Byte Count value is used to set the size of the last HyperTransport data packet.

Command Field		Command Type	Packet Type/ (size in bytes)
Cmd[5:0]	=	Write Request - Nonposted/posted - data length = byte/doubleword - normal/isochronous - noncoherent/coherent	Req/Addr/Data (8)
		Write Request with Extended Address	Req/Addr/Data (12)
		Read Request - Ordering - pass/no pass - data length = byte/doubleword - normal/isochronous - noncoherent/coherent	Req/Address (8)
		Read Response	Resp/Data (4)
		Target Done	Response (4)
		Broadcast Message	Req/Address (8)
		Atomic Read-Modify-Write	Req/Addr/Data (8)
		Address Extension	Address (12)
		Flush posted writes	Request (4)
		Fence for posted requests	Request (4)
		Extended Flow Control for VCSets 0-7	Info (4)
		Link Synchronization and Error Packet	Info (4)

Figure 45: Common HyperTransport command formats

HyperTransport Virtual Channel Packet Format

Bit Time	7	6	5	4	3	2	1	0
0	SeqID[3:2]		Cmd[5:0]=1011xx					
1	Pass PW	SeqID[1]	SeqID[0]/ ReqVC[3]	UnitID[4:0]				
2	Count[1:0]		Compat	Data Error	Chain	Rsv/ReqVC[2:0]		
3	SOM	EOM	Rsv	Format/D =0	Bcount[1:0]		Count[3:2]	
4	Addr[15:8]							
5	Addr[23:16]							
6	Addr[31:24]							
7	Addr[39:32]							

Important Virtual Channel Packet Fields

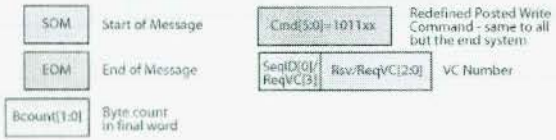


Figure 46: Virtual channel packet format

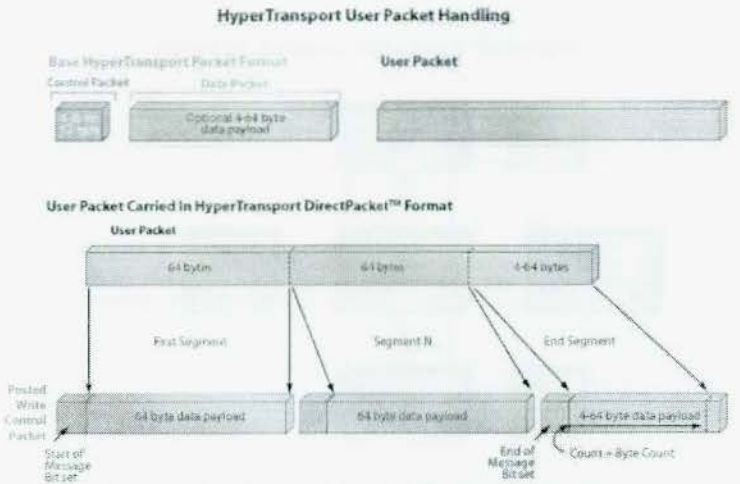


Figure 47: Mapping of user data in HyperTrasnport packets (DirectPacket)

There are two different routing mechanisms specified in HyperTransport: host reflected routing and device-to-device routing.

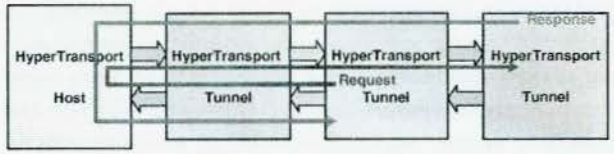


Figure 48: Host reflected routing

Host reflected routing requires that all traffic pass through the host in order to maintain PCI compatibility. To maintain PCI compatibility, the base HyperTransport specification supports host-reflected routing. In this scheme, all traffic passes through the host device in the daisy chain. In the PCI bus structure, since it is a parallel physical bus, it is easy to understand the order in which data transactions take place. In typical processor-based computing systems, the order of transactions is often very important. Since HyperTransport is a distributed system, ordering is more difficult to discern. Thus, in order to replicate the ordering properties of PCI, all base HyperTransport transactions are reflected through the host.

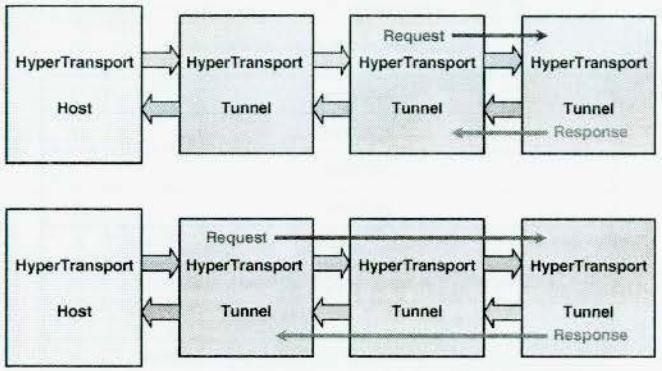


Figure 49: Device-to-device routing

HyperTransport DirectPacket protocols support peer-to-peer routing that allows two devices to communicate directly, greatly reducing link traffic and off-loading the host of the burden of reflecting the device-to-device traffic. This direct device-to-device transaction is typical in complex communications systems that deploy a chain of special function and packet processing engines.

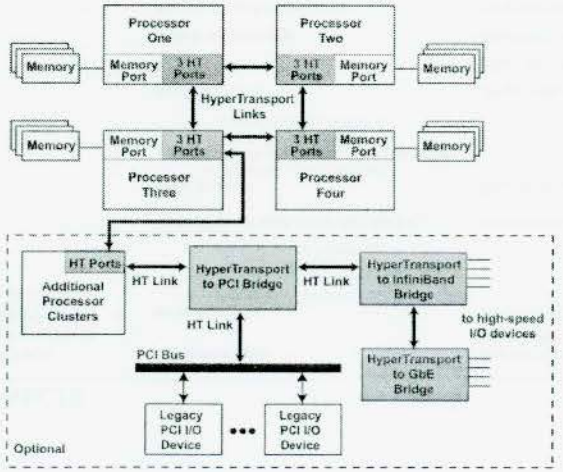


Figure 50: Multiprocessor system application

HT can be used to integrate processor-to-processor, processor-to-memory and processor-to-I/O data streams as shown in this HyperTransport-based multiprocessor system (Figure 50). Bridges can easily link HyperTransport links to legacy or emerging I/O technologies. One continuous HyperTransport link integrates not only interprocessor communication, but brings both low and high bandwidth devices into an integrated I/O stream accessible by all computing elements. Using wide links between processors and narrow links between I/O devices gives the system designer the ability to apply chip-to-chip bandwidth precisely where needed, without having to run expensive links to every node. HyperTransport-enabled architectures also allow the linking of multiple multiprocessor clusters in an efficient, low-cost manner. Through the use of HyperTransport-to-PCI or HyperTransport-to-PCI-X bridges and its full PCI compatibility, HyperTransport technology enables full support of legacy PCI-compatible I/O subsystems without the need for system redesign.

Two other possible applications of HyperTransport in networking and high-performance clustering systems are shown in the figures below.

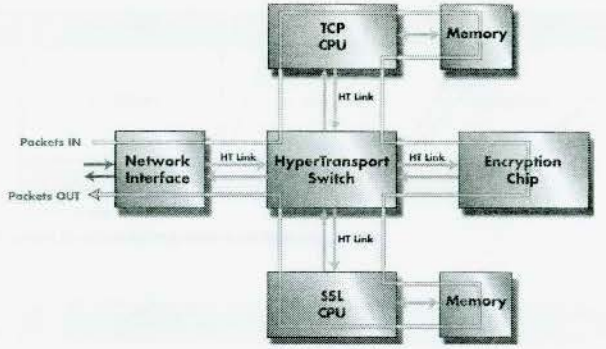


Figure 51: Switched networking application

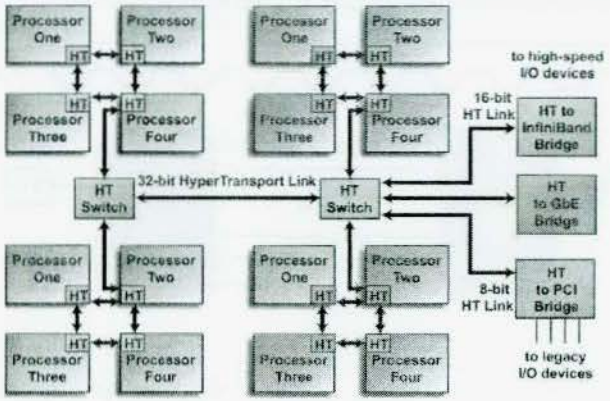


Figure 52: HyperTransport high performance cluster application

1.5 Rapid I/O

The RapidIO architecture is an electronic data communications standard for interconnecting chips on a circuit board and circuit boards using a backplane. This new high-performance, packet-switched interconnect technology was designed for embedded systems, primarily for the networking and communications markets. The RapidIO standard is a packet-switched interconnect architecture conceptually similar to internet protocol (IP). However, the RapidIO architecture is designed to be used for the processor and peripheral interface where high bandwidth and low latency are crucial. The RapidIO architecture is partitioned into a three-layer hierarchy of logical, transport, and physical specifications, which allows scalability and future enhancements while maintaining compatibility.

RapidIO is specified in a 3-layer architectural hierarchy. The logical layer specification defines the overall protocol and packet formats. This is the information necessary for end points to initiate and complete a transaction. The transport layer specification provides the necessary route information for a packet to move from end point to end point. The physical layer specification describes the device level interface specifics such as packet transport mechanisms, flow control, electrical characteristics, and low-level error management. This partitioning provides the flexibility to add new

transaction types to the logical specification without requiring modification to the transport or physical layer specifications.

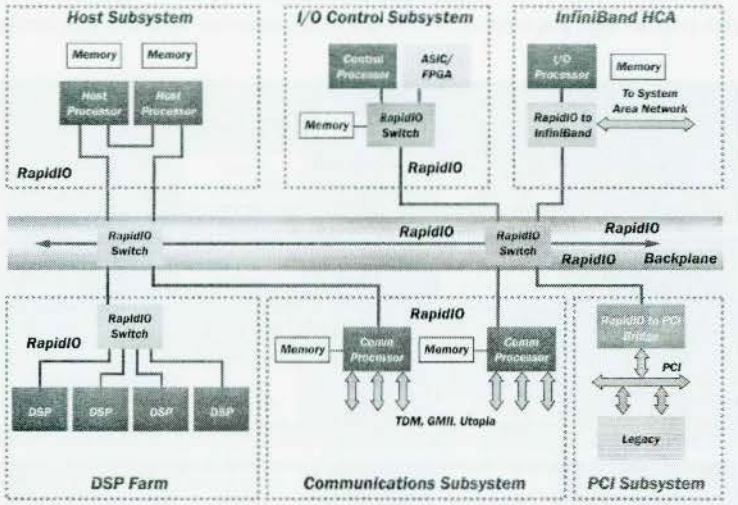


Figure 53: RapidIO interconnect architecture

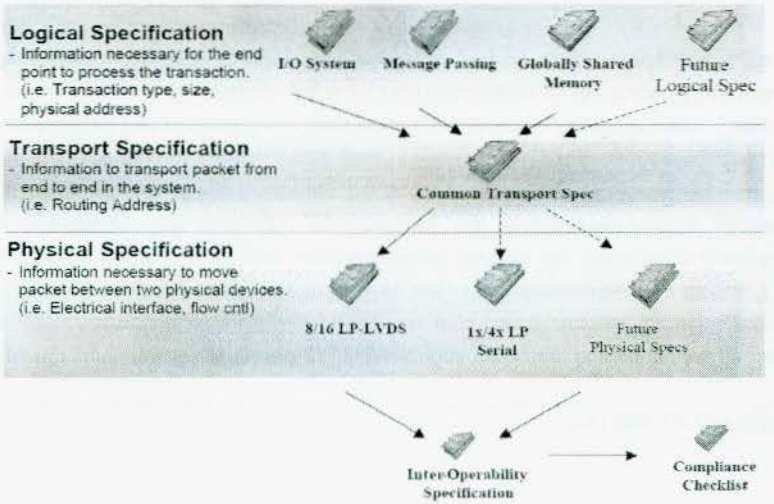


Figure 54: RapidIO specification

The main characteristics of RapidIO are:

- Packet switched: point-to-point interconnect to connect processors, coprocessors, memory, and memory mapped I/O
- Low overhead & low latency; optimized as an "inside-the-box", device level interface
- Small silicon footprint; can be implemented in ASICs and even FPGAs without consuming all of the gates
- Software transparent: an extension of the microprocessor bus that allows direct, physical memory mapping of the entire machine
- Reliable delivery of packets; error detection and recovery in hardware
- Layered architecture which permits varying complexities of switch fabrics and end-points and future additions to the specifications
- Standard I/O technology

The RapidIO logical packet description is defined to be physical layer independent. This means that the RapidIO protocol could be transmitted over anything from serial to parallel interfaces, from copper to fiber media. The first physical interface considered and defined is known as the 8- or 16-bit link protocol end point specification (8/16 LP-LVDS). This specification is defined as having 8 or 16 data bits in each direction along with clock and frame signals in each direction.

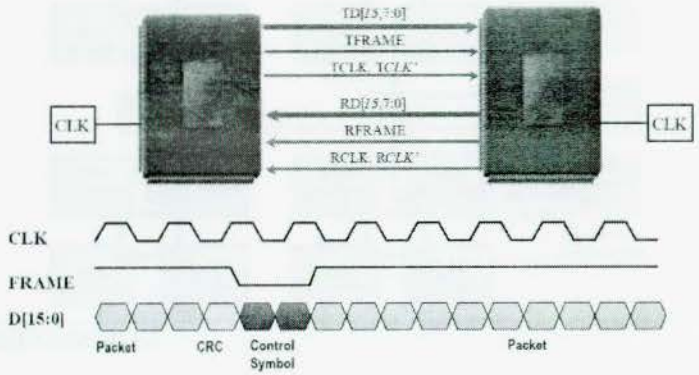


Figure 55: RapidIO 8/16 LP-LVDS physical layer

RapidIO operations are based on request and response transactions. Packets are the communication element between end point devices in the system. Initiator generates a request transaction, which is transmitted to a target. The target then generates a response transaction back to the initiator to complete the operation. The RapidIO transactions are encapsulated into packets, which include all of the necessary bit fields to ensure reliable delivery to the targeted end point. RapidIO end points are typically not connected directly to each other but instead have intervening connection fabric devices. Control symbols are used to manage the flow of transactions in the RapidIO physical interconnect. Control symbols are used for packet acknowledgement, flow control information, and maintenance functions.

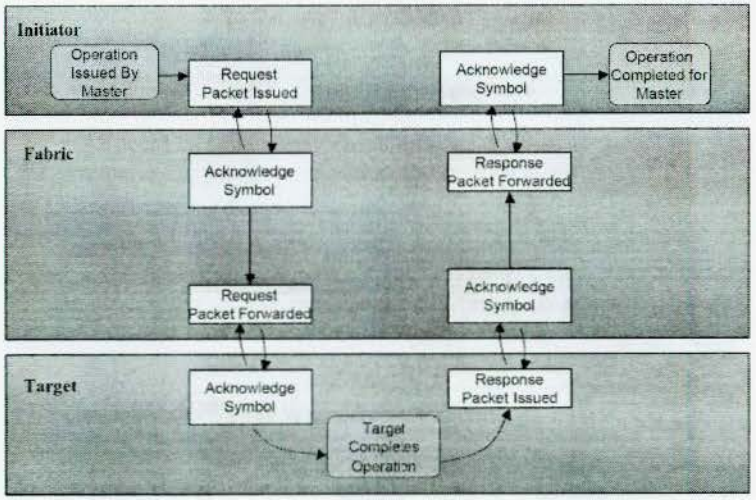


Figure 56: Operation sequence (transactions are constructed with request and response packet pairs)

The RapidIO packet is comprised of fields from the three-level specification hierarchy. Figure 57 shows typical request and response packets. Certain fields are context dependent and may not appear in all packets. The request packet begins with physical layer fields. The "S" bit indicates whether this is a packet or control symbol. The "AckID" indicates which packet the fabric device should acknowledge with a control symbol. The "PRIO" field indicates the packet priority used for flow control. The "TT", "Target

Address", and "Source Address" fields indicate the type of transport address mechanism used, the device address where the packet should be delivered, and where the packet originated. The "Ftype" and "Transaction" indicate the transaction that is being requested. The "Size" is an encoded transaction size. RapidIO transaction data payloads range from 1 byte to 256 bytes in size. The "srcTID" indicates the transaction ID. RapidIO devices may have up to 256 outstanding transactions between two end points. For memory mapped transactions the "Device Offset Address" follows. For write transactions a "Data Payload" completes the transaction followed by a 16-bit CRC.

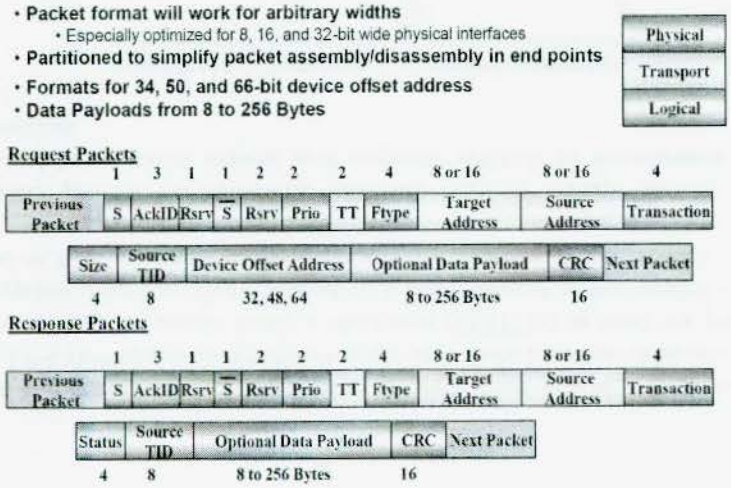


Figure 57: RapidIO packet formats

Response packets are very similar to request packets. The "Status" field indicates whether the transaction was successfully completed. The transaction was successfully completed. The "TargetTID" corresponds to the request packet source transaction ID.

RapidIO protocol is made up of packets and control symbols. Packets provide logical transaction interface between endpoints. Examples are read, write, and message. Packets contain a transport address (source and destination). Control symbols provide the physical layer control for transactions such as acknowledge, retry, end of packet. They also provide

the basis for hardware based error recovery and can be embedded within packets.

RapidIO packet functions:

- I/O non-coherent functions
  - NREAD (read non-sharable memory)
  - NWRITE, NWRITE_R, SWRITE (write non-sharable memory)
  - ATOMIC (read-modify-write to non-sharable memory)
- Port based functions
  - DOORBELL (generate an interrupt)
  - MESSAGE (write to port)
- System support functions
  - MAINTENANCE (read or write configuration, control, and status registers)
- User defined functions
- Cache coherence functions
  - READ (read globally shared cache line)
  - READ_TO_OWN (write globally shared cache line)
  - CASTOUT (surrender ownership of globally shared cache line)
  - IKILL (instruction cache invalidate)
  - DKILL (data cache invalidate)
  - FLUSH (return globally shared cache line to memory)
  - IO_READ (read non-cacheable copy of globally shared cache line
- OS support functions
  - TLBIE (TLB invalidate)
  - TLBSYNC (force completion of TLB invalidates)

Control symbols are a physical layer entity. They are 16 bits long (repeated inverted for robustness) and may be inserted inside of packets Delineated by FRAME

Some examples of control symbols:

- Packet acknowledge (accepted, retry)
- Packet delineation (Idle, End of Packet, Stomp)

- Flow control (Throttle)
- Link status (Link Request, Link Response)

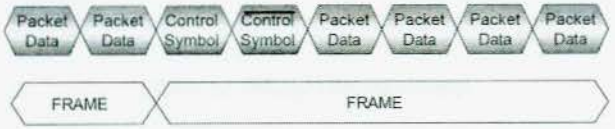


Figure 58: Cotrol symbols

Error management is implemented in the both physical and protocol layer. 8/16 LP-LVDS specification allows single and multi-bit error detection and recovery. At the packet level, a checksum (CCITT16) is used for packet corruption detection. Heavy encoding is implemented at the symbol level. Fixed ACKID sequence numbering is used for detection of lost packets.

RapidIO provide an automatic hardware retry of corrupt packets and physical layer status request and response symbols to maintenance the connection.

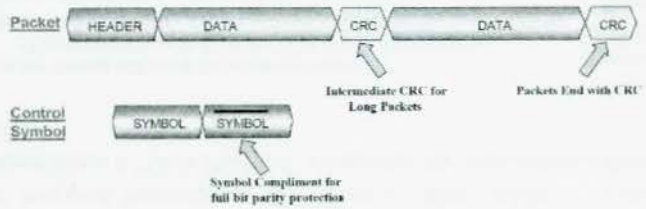


Figure 59: Error management

RapidIO also describes three types of flow control mechanisms; retry, throttle, and credits. The retry mechanism is the simplest mechanism and is required not only for flow control but also as a component of hardware error recovery. A receiver, unable to accept a packet because of a lack of resources or because the received packet was corrupt, may respond with a retry control symbol. The sender will retransmit the packet.

The throttle mechanism makes use of the idle control symbol. Idle control symbols may be inserted into a packet during packet transmission. This

allows a device to insert wait-states in the middle of packets. A receiving device can also send a throttle control symbol to the sending device requesting that it slow down by inserting idle control symbols.

The credit-based mechanism is useful for devices that implement transaction buffer pools, especially fabric devices. In this scheme certain control symbols contain a buffer status field that represents the current status of the receiver's buffer pool for each transaction flow. A sender only sends packets when it knows the receiver has available buffer space. This information enables switch fabric devices to discover downstream blocking conditions and to make better packet output selections.

Special packet formats are defined for system discovery, configuration, and error management. They provide an access to non-memory mapped control and status register space. Register set includes device capabilities, setup, and status information.

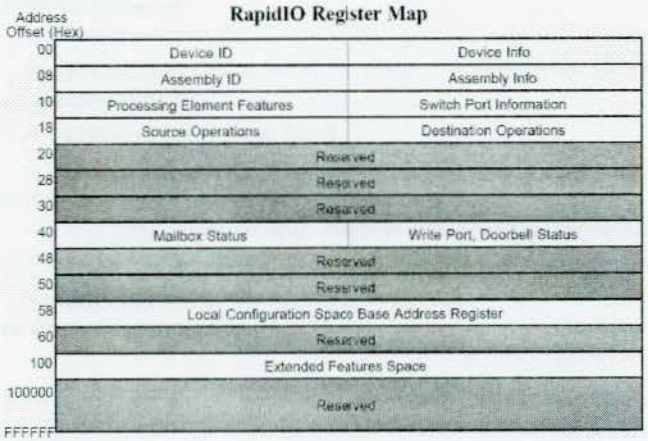


Figure 60: Maintenance

RapidIO supports arbitrary hardware topologies. Discovery mechanism allows software discovery and configuration of devices within the system. Maintenance transaction is used to discover devices. Unique ID register allows detection of already discovered devices. HOP count field in a

maintenance transaction is used to traverse switches. Multiple host concurrent discoveries are supported by the host ownership atomic register.

The efficiency of RapidIO transactions is summarized in Table 3. If sustained operations include all transaction overhead and assume fully loaded full duplex communications then:

- 32-byte operation is 50% efficient
- 256-byte operation is >90% efficient

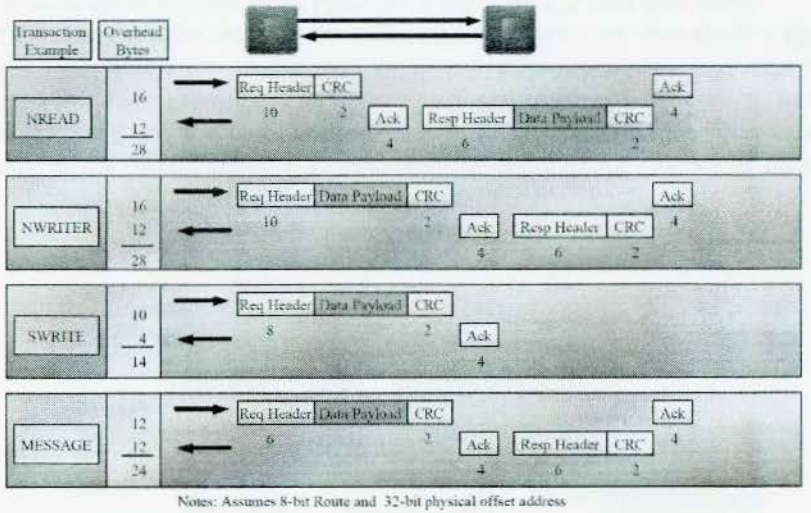
For Unidirectional traffic:

- 32-byte is >70% efficient
- 256-byte is >95% efficient

Configuration	8-bit Mode			16-bit Mode		
	PEAK	Sustained 32 byte Op	Sustained 256 byte Op	PEAK	Sustained 32 byte Op	Sustained 256 byte Op
125MHz	4Gb/s	2Gb/s	3.7Gb/s	8Gb/s	4Gb/s	7.5Gb/s
250MHz	8Gb/s	4Gb/s	7.5Gb/s	16Gb/s	8Gb/s	15Gb/s
500MHz	16Gb/s	8Gb/s	15Gb/s	32Gb/s	16Gb/s	30Gb/s
1GHz	32Gb/s	16Gb/s	30Gb/s	64Gb/s	32Gb/s	60Gb/s

Table 3: RapidIO bandwidth summary

A performance concern in any tightly coupled intra-system interconnect is the transaction overhead required for the interface. Such overhead includes all bytes sent to complete a transaction such as arbitration, addresses, acknowledgements, error coverage, etc. Figure 12 shows some typical RapidIO operations and the number of bytes required to complete each operation. It is important to remember that RapidIO is a full duplex interface and therefore the interface can be fully pipelined with several outstanding operations at various stages of completion. Reply overhead does not contend with sending overhead. This is different from traditional buses, which require turnaround cycles and arbitration phases that add to the overhead.



Notes: Assumes 8-bit Route and 32-bit physical offset address

Figure 61: RapidIO packet overhead examples

1.6 InfiniBand

The InfiniBand Architecture (IBA) is an industry-standard architecture for server I/O and inter-server communication. It was developed by the InfiniBand Trade Association (IBTA) to provide the levels of reliability, availability, performance, and scalability necessary for present and future server systems, levels significantly better than can be achieved with bus-oriented I/O structures.

The smallest complete IBA unit is a subnet, illustrated in Figure 62. Multiple subnets can be joined by routers (not shown) to create large IBA networks.

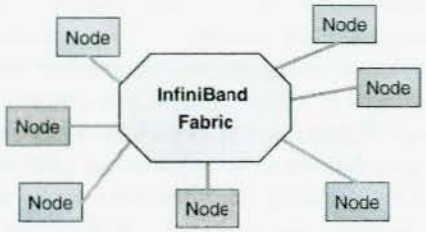


Figure 62: InfiniBand topologies and components

The elements of a subnet, as shown in the figure, are endnodes, switches, links, and a subnet manager. Endnodes, such as hosts and devices, send messages over links to other endnodes; the messages are routed by switches. Routing is defined, and subnet discovery performed, by the Subnet Manager. Channel Adapters (CAs) connect endnodes to links. Links may also incorporate retiming repeaters, but since these are architecturally invisible they will not be mentioned further.

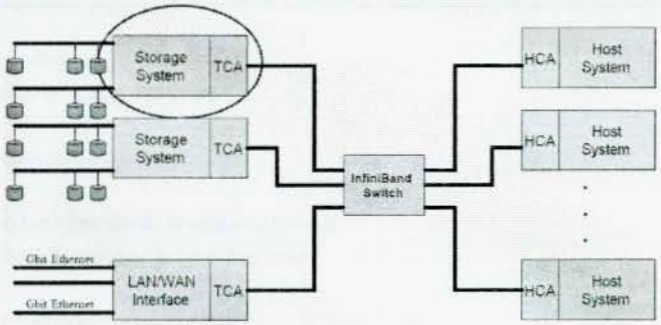


Figure 63: InfiniBand in enterprise storage

The interface between an endnode and a link is a Channel Adapter (CA). Host CAs (HCA) differ from the CA for a device (called a Target CA, TCA) in that the HCA has a collection of features that are defined to be available to host programs, defined by verbs; a TCA has no defined software interface.

The position of an HCA in a host system is vendor-specific. It is expected that initial implementations will provide HCAs as cards attached to a standard I/O bus in order to quickly provide platforms for software

development and evaluation. Ultimately, however, HCA implementations will undoubtedly attach directly to, or become a part of, the host memory control subsystem and partly or completely replace current I/O busses. CAs source the several communication service types of IBA, using queues to hold requests for work to be done and completions. HCAs also contain a specific memory model.

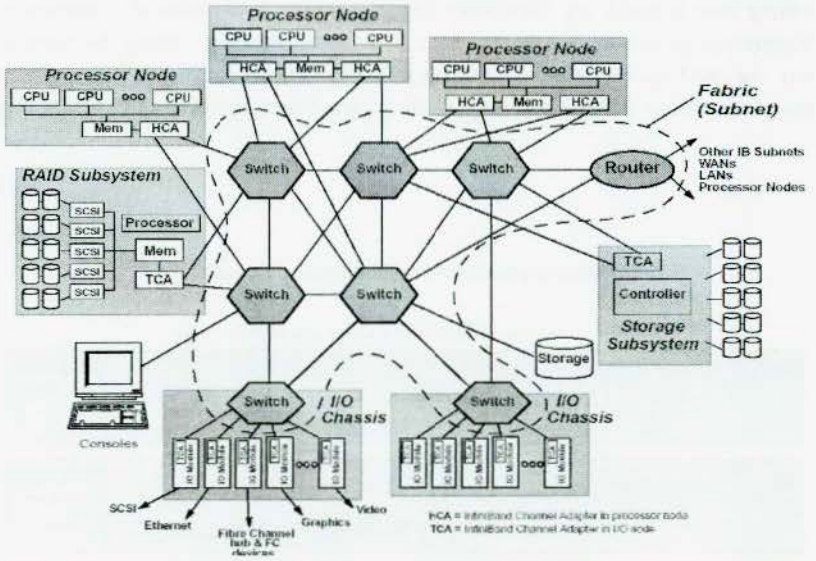


Figure 64: An InfiniBand system area network

For all the service types, CAs communicate using Work Queues of three types: Send, Receive, and Completion. Send and Receive Queues are always used as Queue Pairs (QPs), as illustrated in Figure 66. A particular QP in a CA is the destination or source of all messages: Connected services connect QPs, and unconnected services target QPs on other CAs, specifying a QP number along with the Local Identifier (LID) of the CA port they target.

Queuing:

- Consumer (user) queues up a set of instructions for hardware to execute (Work queue)

- Work queues are created in pairs (Queue pairs – QP) for send and receive operations
- Each Work Queue has corresponding Completion Queue

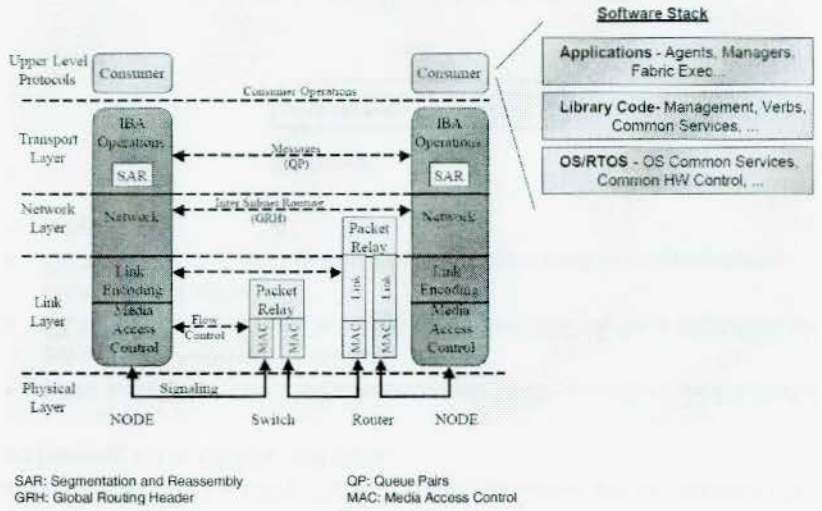


Figure 65: InfiniBand architecture

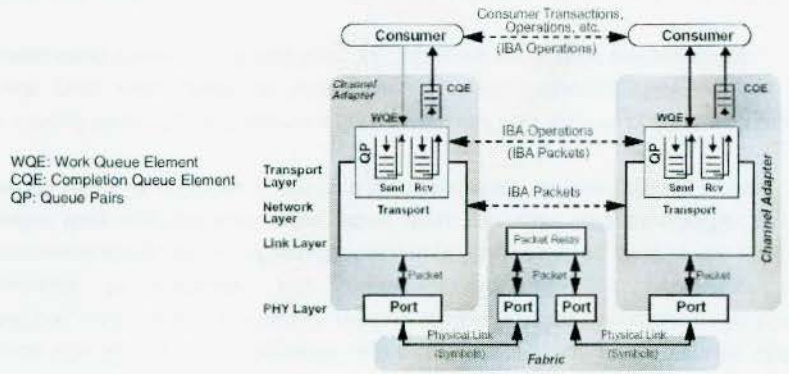


Figure 66: InfiniBand communication stack

An InfiniBand channel adapter (CA) is a programmable DMA engine with special protection features that allow DMA operations to be initiated locally

and remotely. Host Channel Adapter (HCA) provides a consumer interface providing the functions specified by IBA verbs. Target Channel Adapter (TCA) provides an interface to the device

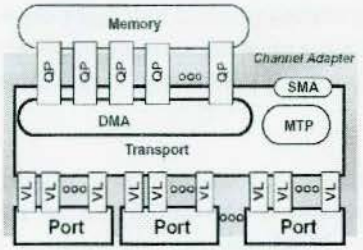


Figure 67: Channel adapter

The type of service used and QP's size, the maximum number of requests that can be queued, are specified when a QP is created. Each QP also has an associated port, also specified when the QP is created; this is an abstraction of the connection of a CA to a link. CAs may have multiple ports, as also illustrated. The number of QPs is at least three: Two for management, and one for operation. The smallest maximum size of a QP (number of queued requests) is vendor-specific. Quality of Service (QoS) is supported by virtual lanes (VLs). Memory Translation & Protection (MTP) mechanism is used to translate virtual addresses to physical addresses and validate access rights.

A subnet manager (SM) is an entity attached to a subnet responsible for its management. Responsibilities of a subnet manager are to discover topology as well as to configure the channel adapter port with a range of local and global IDs (LID, GID), subnet prefix and Partition_Keys. It also maintains LID/GID resolution tables.

IBA management is defined in terms of managers and agents. Managers are active entities; agents are passive entities that respond to messages from managers. The only exception to agent passivity is that they may optionally emit traps targeting managers.

Every IBA subnet must contain a single master subnet manager, residing on an endnode or a switch. It discovers and initializes the network, assigning

Local IDs (LIDs) to all elements, determining path maximum transfer units (MTUs), and loading the switch routing tables that determine the paths from endnode to endnode. The master subnet manager does this by communicating with Subnet Management Agents that must exist on all nodes; they respond with information about the node, such as whether it is a switch or a CA, whether it can be a manager, if so what its priority is, etc.

A virtual lane (VL) represents a set of transmit and receive buffers in a port. Each port must have at least one data VL. Separate flow control is maintained over each virtual line. VL15 is used for subnet management.

Each end-node has one or more Channel Adapters (CAs) and each CA has one or more ports Each Queue Pair (QP) has a QP number (QPN) assigned by the CA. Each port has a unique 16-bit Local ID (LID) and at least one IPv6 address (Global ID – GID). Three types of quantities are important to IBA addressing: LIDs, GUIDs, and GIDs.

- LIDs, Local Identifiers, are subnet-unique 16-bit identifiers used within a network by switches for routing.
- GUIDs, Global Unique IDs, are 64-bit EUI-64 IEEE-defined identifiers for elements in a subnet.
- GIDs, Global IDs, are 128-bit global identifiers used for routing across subnets.

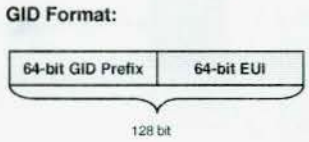


Figure 68: Format of the global identifier (GID)

Several examples of IBA packets are illustrated in Figure 69. There are packets that use IBA transport for local (intra-subnet) or for local (inter-subnet) traffic; and provision for "raw" packets that don't use the IBA transport layer for both of those cases 8not shown in the figure). The raw format is intended to simplify processing in routers that do not target other IBA subnets, allowing hosts to directly provide them with packets already in the format needed. Which of those formats is present in a packet is indicated

by the Link Next Header (LNH) field in the initial Local Routing Header that is always present, as discussed below.

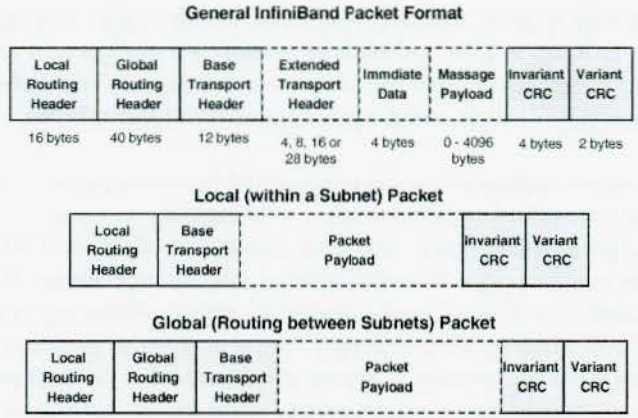


Figure 69: InfiniBand packet format

The complete format of IBA packets is shown in Figure 69. This general InfiniBand packet format has fields that are optional or may not be present are indicated by dashed lines (the Base Transport Header is not optional when IBA transport is used). The intimate details of all the bits in the headers can be found in the InfiniBand architecture specification. The general purpose and approximate contents of the fields (not all elements are described) are:

- Local Routing Header (LRH): This is used by switches to move a packet across a subnet to its destination endnode network port. It contains the source and destination LIDs, link protocol version ID, service level, virtual lane, packet length, and a "next header" field used to indicate which optional fields are present.
- Global Routing Header (GRH): This is used by routers to move packets between subnets. In addition to source and destination GIDs and global routing version ID, it contains payload length, IP version, traffic class, and hop limit. The LRH of a globally routed packet moves it to a router, which uses the GRH to determine where it goes; the destination router constructs a new LRH to move the packet to its destination port over the target subnet.
- Base Transport Header (BTH): This tells endnodes what to do with

packets. In addition to yet another version ID, it has an opcode, destination Queue Pair, packet sequence number for reliable transports, partition key, and a number of other fields. Whether extended headers are present and what kind they are is indicated by the opcode.

- Extended Transport Header (ETH): This holds additional information depending on the BTH operation or LRH next header. For example, for RDMA operations it has a virtual address, length, and R_Key for atomic operations, it has an opcode, virtual address, data to be swapped or compared, and an R_Key again; for ACKnowledgement packets it contains a syndrome and a sequence number; etc.
- Message Payload: This is the point of the whole exercise in most cases.
- Invariant CRC: This is a CRC over all the fields of the packet that cannot change in transit. It is not present in raw packets.
- Variant CRC: This is a CRC over all fields of the packet, including the Invariant CRC. It is present because, for example, the LRH must be replaced in globally routed packets.

1.7 Common Switch Interface (CSIX)

The common switch interface (CSIX), is a detailed interface specification between port/packet processor logic and interconnect fabric logic. It is a scalable parallel interface with separate data and control paths.

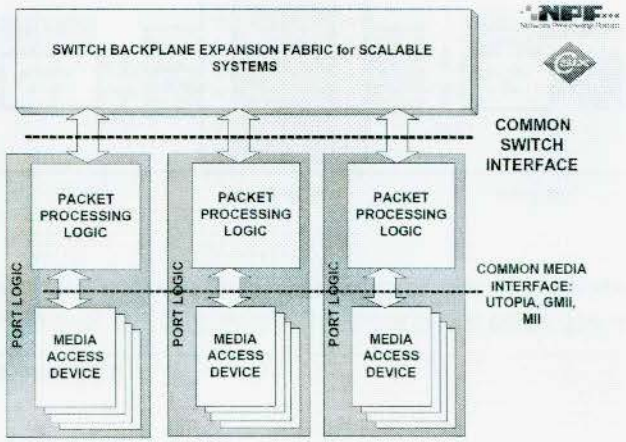


Figure 70: Positioning of common switch interface (CSIX)

CSIX is a generic multi-vendor specification is established to promote the deployment and development of highly scalable network switches, to permits hardware and software interoperability and a mix and match of interchangeable silicon components, hardware and software. This concept can be used to expand existing switch architectures: point-to-point, shared memory, and shared bus.

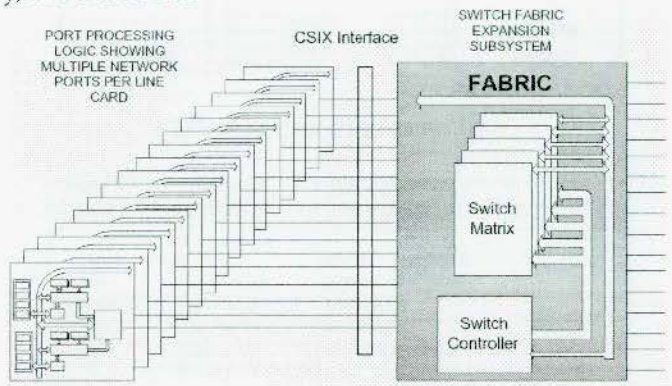


Figure 71: Fabric - point of concentration

The level 1 specification of the common switch interface (CSIX-L1) can be used to connect multiple traffic managers to a switching fabric to create the core of a switch design as shown in Figure 72. Some switching fabric vendors may offer modular fabric systems that support connectors within the fabric. When used with such a fabric system, CSIX-based modular designs can be created by combining one or more traffic managers on a line card with a portion of the switching fabric.

Traffic managers (TMs) shown in the figure represent a logical entity performing collection of functions (e.g., fragmentation and reassembly, traffic shaping and security processing) implemented by a switch fabric user to manage the flow of data between a number of ports (line ends) and the switch fabric.

CSIX-L1 supports up to 4096 CSIX ports for connecting traffic managers to the fabric. The number of ports is determined by the fabric and is vendor-specific. The CSIX 8-bit class variable provides a mechanism to manage traffic flows between a traffic manager and a fabric port. Classes can be used

to manage priority and QOS services provided by the fabric. The number of classes supported is also determined by the fabric and is vendor-specific.

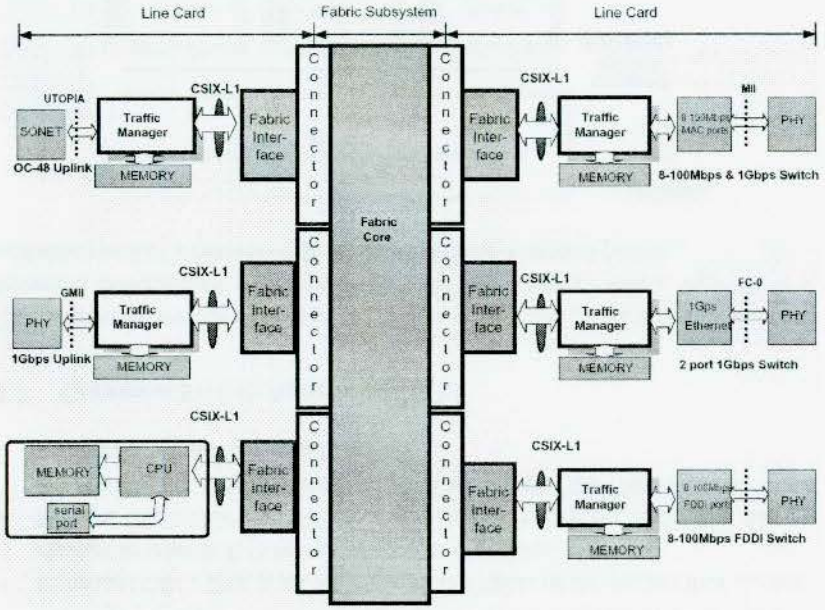


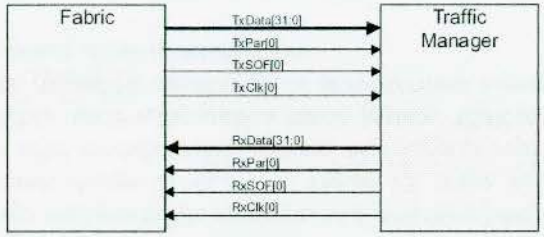
Figure 72: Common switch interface level 1 (CSIX-L1) implementation

There are three levels in CSIX-L1, namely logical, interconnection and physical level.

Logical or Message Level
This level ensures that data or control message protocol exchanged over the interface are properly understood by each end and properly processed by the appropriate function.

Interconnection Level
It defines all the signals with specific functions, meanings, and bit widths, input or output, signal handshake protocols, etc.

Physical Level
The physical level specifies the electrical characteristics such as voltage levels, capacitance, drive strengths, timings etc.



Signal	Direction	Function
RxData[31:0]	TM to Fabric	Receive Data
RxPar[0]	TM to Fabric	Receive Data Odd Parity RxPar[0] -> RxData[31:0]
RxSOF[0]	TM to Fabric	Receive Start of Frame
RxClk[0]	TM to Fabric	Receive Clock
TxData[31:0]	Fabric to TM	Transmit Data
TxPar[0]	Fabric to TM	Transmit Data Odd Parity TxPar[0] -> TxData[31:0]
TxSOF[0]	Fabric to TM	Transmit Start of Frame
TxClk[0]	Fabric to TM	Transmit Clock

Figure 73: CSIX-L1 32-bit interface

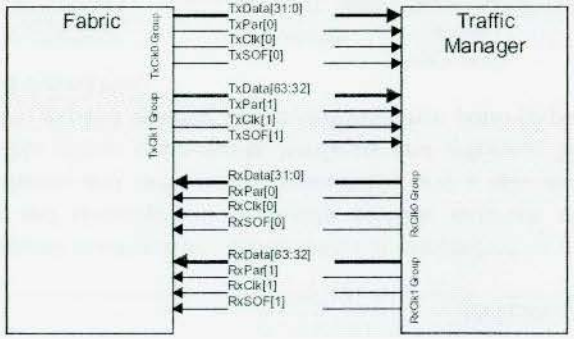


Figure 74: CSIX-L1 64-bit interface

CSIX-L1 utilizes an $n \times 32$-bit data path, where $n = 1, 2, 3$ or 4. The frequency of operation for CSIX-L1 is specified for up to 250 MHz. Figure 73 shows a 32-bit interface with description of its signals. Transmit signals carry information from the switch fabric to the traffic manager; receive signals carry information from the traffic manager to the switch fabric.

The 64-bit and 128-bit CSIX-L1 interface specifications are shown in Figures Figure 74, Figure 75, and Figure 76. The extension is easily made by adding one or three additional 32-bit interfaces to the standard 32-bit interface. The 32-bit busses are grouped into transmit and receive groups, each with same command signals as in the 32-bit specifications and with separate transmit and receive clock signals for each group.

Signal	Direction	Function
RxData[63:0]	TM to Fabric	Receive Data
RxPar[1:0]	TM to Fabric	Receive Data Odd Parity RxPar[0] -> RxData[31:0] RxPar[1] -> RxData[63:32]
RxSOF[1:0]	TM to Fabric	Receive Start of Frame
RxClk[1:0]	TM to Fabric	Receive Clock RxClk[0] Group: RxData[31:0], RxPar[0] and RxSOF[0] RxClk[1] Group: RxData[63:32], RxPar[1] and RxSOF[1]
TxData[63:0]	Fabric to TM	Transmit Data
TxPar[1:0]	Fabric to TM	Transmit Data Odd Parity TxPar[0] -> TxData[31:0] TxPar[1] -> TxData[63:32]
TxSOF[1:0]	Fabric to TM	Transmit Start of Frame
TxClk[1:0]	Fabric to TM	Transmit Clock TxClk[0] Group: TxData[31:0], TxPar[0] and TxSOF[0] TxClk[1] Group: TxData[63:32], TxPar[1] and TxSOF[1]

Figure 75: CSIX-L1 64-bit interface signals

A CFrame is the base information unit transferred between traffic managers (TMs) and a CSIX Fabric. It consists of a base header, an optional (determined by type) extension header, an optional payload, optional padding bits and a 16-bit vertical parity field. The payload is variable in length and is passed by the CSIX Fabric from ingress traffic manager to egress traffic manager.

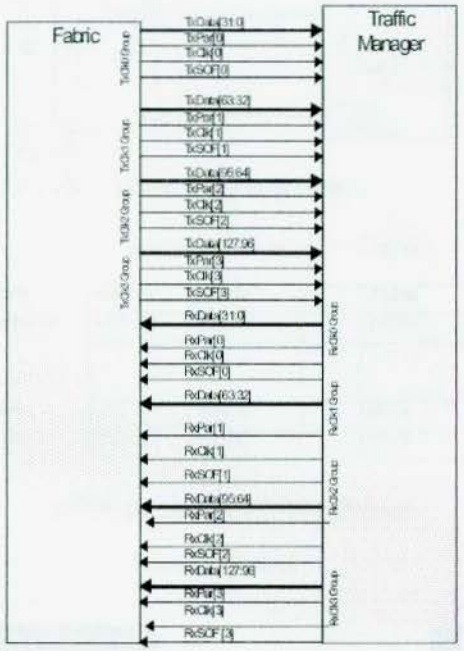
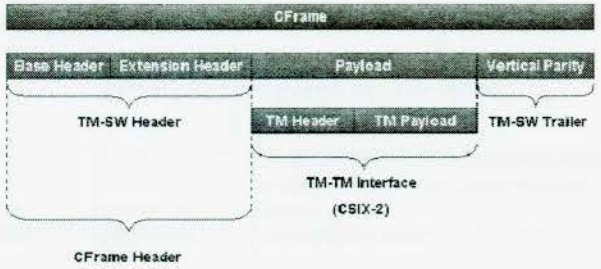


Figure 76: CSIX-L1 128-bit interface

When the TM-TM Interface message size is greater than the maximum CFrame Payload size, multiple CFrames must be transferred from ingress to egress TM in order. An ordered collection of CFrames is defined as a CFrame Sequence. In each of the CFrames the TM-TM Interface will contain the necessary TM Header and TM payload information needed to reassemble the TM-TM Frame at the egress TM.

Vertical parity provides an optional second set of parity bits that provide a substantial improvement (over just horizontal parity) in the probability of error detection. To calculate vertical parity bits, the CFrame is treated as a series of 16-bit words, organized in a two-dimensional block. A Vertical parity bit is generated for each of the 16 bit positions (columns) in the block across all rows required to contain the CFrame payload. The resulting 16-bit Error Detecting Code (VPar) is appended to the end of the payload.



CFRAME COMPONENT	Base Header	Extension Header	Payload	Vertical Parity
LENGTH	2 bytes	0-4 bytes	Maximum allowable length is 256 bytes.	2 bytes
COMMENTS		Number of bytes determined by CFrame and address type	Features of the traffic manager and fabric determine actual maximum length.	Field is required; use of field is optional.

Figure 77: CFrame structure

The pause and resume information (carried with Ready bits in the base header of every CFrame) provides a way to control the flow of data across CSIX. De-assertion of a Ready bit indicates that the asserting device can not accept another frame after the one currently being transferred for that link level queue. The device receiving the pause shall complete the current frame and shall not resume transmission of CFrames until an asserted ready bit is received indicating a resume.

Response Requirement:

From the tick that a frame containing a Pause status leaves one component, after maximum response ($n \cdot T$) time has elapsed, no new frame can begin transmission between the components for the congested link level maximum response time for a Pause operation is defined as:

$$\boxed{T_{max} = n \cdot T; \quad n = C + L},$$

where T is the clock period of the interface, n is the maximum number of ticks for the response, and C is a constant for propagating the field within the "other" component (or chipset as the case may be) to the interface logic

controlling the reverse direction data flow. C is defined to be 32 ticks. L is the maximum number of ticks to transport the maximum fabric CFrame size.

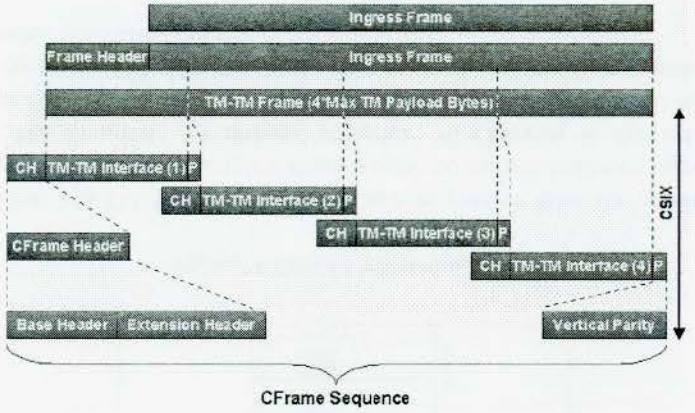


Figure 78: CFrame sequence example

Word 0	B(15,0)	b(14,0)		B(0,0)
Word 1	B(15,1)	b(14,1)		B(0,1)
Word m	B(15,m)	b(14,m)		B(0,m)
VPar	*VPar15*	*VPar14*		*VPar0*

$VPar[i] = !(b(i,0) \wedge b(i,1) \wedge ..... \wedge b(i,m));$

Figure 79: Vertical parity

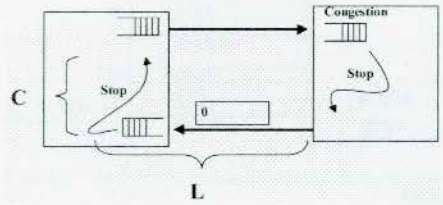


Figure 80: Pause and resume in common switch interface (CSIX)

1.8 Backplanes

An increasingly broad set of systems uses backplanes to connect boards including telecom switches, multi-service provisioning platforms, add/drop multiplexers, digital cross connects, storage switches, routers, embedded platforms, multiprocessor systems and blade servers. These systems typically provide modularity through the addition of line cards, switch cards and services blades to a single high-speed backplane. In most of those platforms the backplane uses a serial interconnect operating at rates higher than 1 Gbit/s.

Communication and computing systems share some common backplane requirements. Because of legacy space constraints, the size of the backplane-based chassis is around 60 cm. The backplane interconnect is often specified to operate over distances of up to 1 meter with two connectors.

Backplane trace lengths may be different, introducing different delays and requiring skew compensation. Most modular systems will block data at some point, such as the switch card or the egress line-card port that may be the target of more traffic than it can handle. To address this congestion, backplanes are designed to perform flow control and traffic management. However, backplane-based communication and computer systems also have unique requirements. Telecom systems are required to supply redundancy, failover mechanisms and bit error rates (BER) as low as 10-15, which in turn drives a backplane BER of better than 10-17. Ethernet has traditionally offered a BER of 10-12, requiring technologies like forward error correction to enhance Ethernet for telecom backplane applications. By contrast, storage switches are often defined by their need for very low latency. For blade-server applications, interprocessor communication may also require low latency.

1.8.1 Classical Backplane Solutions

To implement a backplane almost all interconnecting technologies can be used. Some of the technologies suited fo bus and switched topologies are listed below.

Backplane Busses or Interconnects

- All LVTTL/LVCMOS
- LVDS
- RocketIO or similar transceiver
- ISA
- CompactPCI (3.3V Signaling)
- PCI-X (3.3V Signaling)
- PCI Express
- VME

Switched Backplane

- SpaceWire using LVDS, RocketIO or similar
- RapidIO
- Serial RapidIO
- Ethernet
- InfiniBand
- Star Fabric
- Advanced Switching

Classical backplane circuit boards contain sockets or expansion slots for connections to other circuit boards. There are two types of backplanes: active and passive. Active backplanes contain bus control and bridges; however, they do not contain processor complex components such as the central processing unit (CPU), chipset, or cache. Passive backplanes contain circuitry for bus connectors and, in some cases, buses and drivers.

Several types of buses are used in classical backplanes. The most used are PCI and ISA. Industry standard architecture (ISA) buses are I/O devices that can handle 16-bit data transfers at a clock speed of 8 MHz. Extended ISA (EISA), an enhanced version of the ISA bus, is capable of 32-bit data transfers. Peripheral component interconnect (PCI), a local bus system for high-end computers, can transfer 32 or 64 bits of data at a clock speed of 33 MHz. Compact PCI (cPCI) uses the electrical standards of the PCI bus, but is packaged in a Versa Module Eurocard (VME) bus. The VME bus (VMEbus) is a rugged, 32-bit device used in industrial, commercial and military applications. VME64 is an expanded version of the VMEbus that provides 64-bit data transfers. Other bus types include VME extensions for instrumentation (VXI), multisystem extension interface (MXI), embedded technology extended (ETX), and embedded PCI extended (ePCI-X). Advanced telecommunication computing architecture (AdvancedTCA) is an open specification for carrier grade equipment. Host media processing (HMP) performs the media processing functions of communications applications without special-purpose telephony hardware.

There are many form factors for backplanes. Advanced technology (AT) is the original IBM motherboard design for personal computers (PCs). Full-size

AT boards are up to 12" wide by 13.8" deep. By contrast, small or "baby" AT boards are only 8.57" wide by 13.04" deep. Low profile (LPX) backplanes mount expansion slots on a bus riser that connects to the motherboard. ATX combines features from the AT and LPX designs. Related technologies include MiniATX and microATX. NLX, another form factor for backplanes, supports both current processors and technologies such as accelerated graphics port (AGP), universal serial bus (USB), and dual in-line memory module (DIMM). NLX cards are up to 9" wide and 13.6" deep.

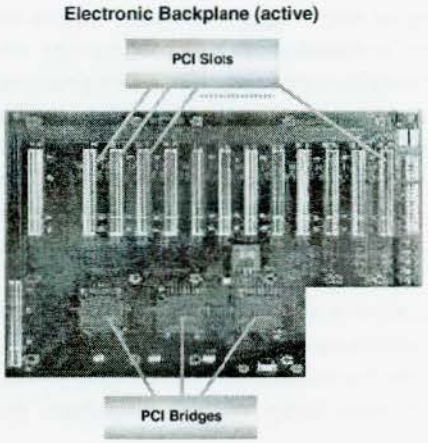


Figure 81: Electronic backplanes

Important specifications for backplanes include the number of segments in the card and the number of expansion slots. Typically, active backplanes include a slot for plugging in the CPU. The bus type determines whether the backplane contains slots for ISA, PCI, cPCI, PXI, VME, VXI, or MXI slots. The form factor determines whether backplanes support AT, ATX, terminal block, or 3.3 V power connectors. Some backplanes include a connector for a keyboard. Others include on-board terminators that minimize reflections and interference on the bus.

1.8.2 High-Capacity Backplanes

High-speed backplanes in modular chassis-based systems, such as core routers, Ethernet switches and storage subsystems, require high levels of signal integrity and increased system throughput.

Today, backplanes in some of these systems are operating with 5 Gbit/s and higher speed serial link technology. To design highly reliable systems at these data rates requires chip vendors to deliver solutions that guarantee error-free transmission in backplanes.

As data rates increase beyond the 1 Gbit/s level, designers must address new problems in their backplane system design. The signal integrity of these backplanes is affected by skin effect, dielectric loss, increased noise due to crosstalk and intersymbol interference (ISI).

Skin effect is a phenomenon where majority of the current flow is concentrated on the outer conductor as frequency increases. The loss due to skin effect is proportional to the square root of frequency, width and height of the trace.

Dielectric loss is due to the heat lost to the board dielectric, and increases linearly with frequency. At higher frequencies, dielectric loss becomes a bigger issue. These losses reduce the signal amplitude, slowing down signal edge rates that can cause signal dispersion and poor jitter budget.

The dispersion of signal leads to ISI as the less attenuated low frequency components are summed up with the attenuated high frequency components at the receiver. As a result, the eye opening gets smaller, making it difficult to recover at the receiving end and can lead to unacceptable bit errors. This limits the maximum bit rate. Another way of explaining this phenomenon is that the signal gets smeared such that the energy from one bit degrades into subsequent bits causing bit errors. Since there is sufficient timing margin at lower rates, the ISI can be corrected. At higher rates, however, ISI is no longer limited to signal boundaries and can affect the whole width of the bit.

Major source for noise is crosstalk that results from the placement of high-density connector and backplane traces. There are two types of crosstalk: near-end crosstalk (NEXT) and far-end crosstalk (FEXT). NEXT is caused when a received signal is corrupted by a signal from the transmitter located next to the victim receiver. FEXT occurs when a received signal is corrupted by the transmitter that is located near the far-end transmitter connected to the victim receiver. All channel impairments are compensated or eliminated in backplane interconnect devices with special signal-conditioning circuits. These circuits attenuate low frequencies and amplify high frequencies to compensate for the loss.

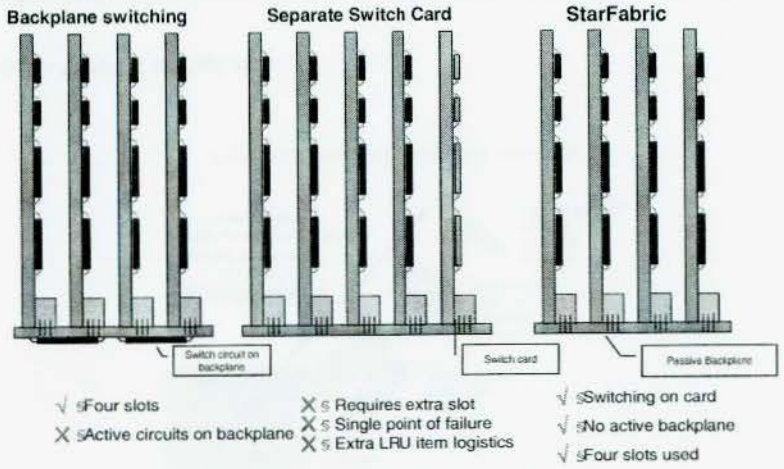


Figure 82: Switched backplane solutions

There are a wide variety of communication systems developed today that share common topologies. In chassis based systems, there are typically several line cards and switch cards interconnected through a passive backplane. In some cases, separate control cards will be used which must communicate with all the line cards and switch cards. The control devices may also be located on the switch cards.

To avoid single point of failure, switching function can be distributed over the line cards as shown in Figure 82. Thus, in this case each card implements a switching element, and therefore, no active backplane is needed. A

drawback of this solution is that line cards become more complex and more costly.

Advanced switching should provide a standard backplane interface for line cards, switch cards and control cards used in communication system applications. One possible implementation could be to use an industry standard (e.g. Advanced TCA) chassis platform, which allows cards from different vendors to be mixed and matched within a system. This will eliminate the interoperability problems we have today where, for example, 10 Gigabit backplanes may be implemented by using different technologies such as CSIX, PCI Express, InfiniBand, SPI-4, or 10 Gigabit Ethernet (see Chapter 0) interfaces, which are not functionally compatible with each other.

Figure 83 shows an example of backplane implementation by using high-speed Ethernet transceiver modules. In this example, backplane connections are duplicated (plane "A" and plane "B") in order to make the backplane more reliable. Here, the backplane is electrical and the transmission rate is limited to 6.25 Gbit/s per direction.

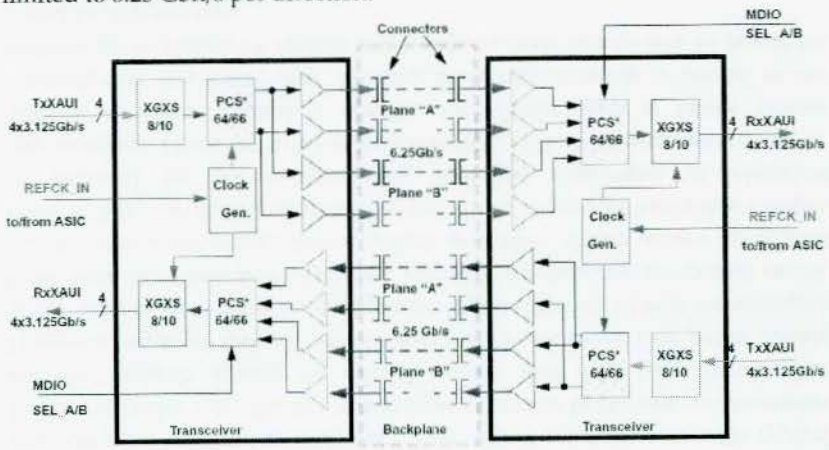


Figure 83: An example of high-speed Ethernet backplane implementation

There is a variety of such modules available on the market (see Figure 84). They are usually equipped with a 10 Gigabit attachment unit interface (XAUI) (see Section 12.3 for more details) and designed for an optical

backplane connection. There is a trend to produce these modules with a small form factor and pluggable in order to achieve an easy use, high density, low power, and low cost. For very high capacity backplanes a solution that uses wavelength division multiplexing (WDM) to transmit many channels over single optical fiber could be convenient, even though such transceiver modules are relatively expensive.

Figure 84: Box-to-box optical modules

1.8.3 Optical Backplanes

Today, optical interconnections in the enterprise are mostly used for the box-to-box interconnects. The optical transceivers are either plugged into high-capacity Ethernet or FC-based switches, or placed on a network interface card (NIC) or a host bus adapter (HBA).

Board-to-Board Interconnects

The move of the communications industry, both telecom and datacom toward a Modular Communication Platform (MCP) favors the deployment of bladed architecture. Functions that were traditionally housed in a standalone box, such as between servers and switches, have started to be implemented into a board-level form-factor, which is called a blade, and they plug into a common chassis. Therefore, interconnect opportunities at the board level are becoming more important. As the transmission speed increases, copper-based interconnects are facing technical challenges in terms of speed, reach, electromagnetic interference (EMI), and routing.

Backplanes are potential applications for optical interconnects. These are point-to-point or point-to-multipoint high-speed interconnects with typical lengths of under 1 m. The key advantages of optical backplane interconnects are low-crosstalk among the optical signals, and their large bandwidth. However, most of today's optical backplanes are more like patch panels rather than replacements for backplanes. Many different optical technologies have been demonstrated including polymer waveguides integrated on Si, planar light wave circuit interconnects, and fiber ribbon arrays integrated with VCSELs and photodiodes. The transition to optical backplanes might be induced by the accumulating technical challenges of electrical interconnects. However to be widely adopted, optical interconnects must be able to advance toward a smaller form-factor with a lower power consumption at a lower cost. Meeting these requirements is critical to the technology evolution of optical interconnects from box-to-box to board-to-board to chip-to-chip.

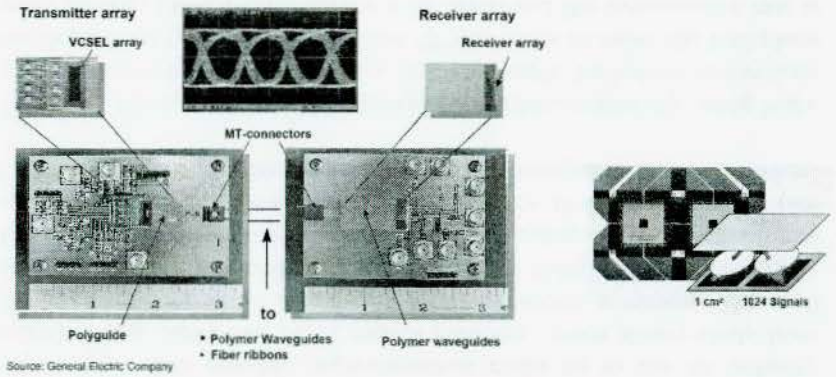


Figure 85: Optical interconnects

Chip-to-Chip Interconnects

It has been shown that frequency-dependent loss for copper traces on FR4 printed circuit boards (PCBs) rapidly rises above 1 GHz, reducing the signal-to-noise ratio (SNR) and introducing timing errors. In addition, the resultant high-frequency crosstalk among the different copper traces limits the wiring density on the circuit board. High-speed short-distance (L < 10 cm) chip-to-chip optical interconnects have several advantages over copper interconnects. Optical interconnects are low-loss interconnects with a large

transmission bandwidth. Another key advantage is their inherent immunity to EMI. The density of copper traces printed circuit boards is constrained by these EMI and electrical crosstalk problems. Over the last 20 years, many different optical technologies were demonstrated to overcome the electrical bottleneck. However, their relative high costs due to implementation complexity and use of exotic materials made them unsuitable for high-volume manufacturing and thus prevented the adoption of these technologies.

On-Chip Interconnects

The design of on-chip electrical interconnects is becoming increasingly difficult given the continuous growth in the complexity of integrated circuits operating at multi-GHz. Can on-chip optical interconnects potentially solve these issues? On-chip point-to-point and point-to-multipoint optical interconnects with typical lengths under 1 cm are potentially attractive because of the following reasons:

- they decrease the existing electrical interconnect delays,
- provide a higher bandwidth to keep pace with the speed of transistors,
- reduce electrical power consumption, and
- minimize sensitivity to EMI.

The potential primary application of on-chip optical interconnects in microprocessors, for example, are in high-speed signaling and clock distribution. For on-chip signal distribution, four key benchmark parameters are typically used: signal delay normalized by clock cycle, available bandwidth per unit area or bandwidth density, bandwidth density/delay ratio, and cost. For on-chip clock distribution, the critical parameters are timing, skew, and jitter. The primary challenges for implementing these optical interconnects is the integration of multiple VCSEL and photodiode arrays with their corresponding drivers and transimpedance amplifiers (TIA), and on-chip light coupling into optical waveguide arrays over the entire chip. Currently, such optoelectronic integration is not only in its early stages of development, but it is also very expensive relative to copper-based interconnects with limited or no performance advantages. In order to take advantage of on-chip optical interconnects; today's microprocessor architecture might need to evolve from a single superscalar chip to a mesh of optically interconnected processors with their associated memories. The WDM scheme could be used, for example, to send multiple wavelengths in

the same optical waveguide to significantly increase the overall communication bandwidth.

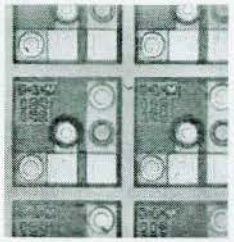


Figure 86: Arrays of vertical cavity surface emitting lasers (VCSELs)

For optical interconnects there are various approaches using different transmission media such as optical fiber, optical waveguide, and free-space.

The spectrum of fiber-based optical interconnects reaches from individual fibers or fiber ribbons, over 2D fiber arrays or fiber image guides, or complex fiber-arrangements that are laminated into flexible foils or even into boards.

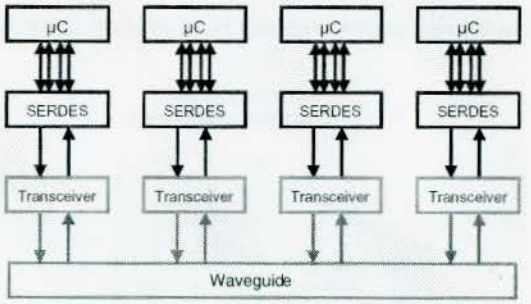


Figure 87: Optical backplanes

Many waveguide-based approaches have been demonstrated in the last few years. Waveguides are fabricated directly into boards or separately onto foils that are laminated into boards subsequently. The waveguide structure itself is defined by process such as direct laser writing; mask exposure, embossing, ion exchange, or etching. The high level of integration within the board is clearly a very interesting feature of waveguides, although this

certainly adds quite some complexity to the manufacturing. Another unique feature is the possibility to directly integrate passive optical elements such as splitters, combiners, or even gratings, which is for example interesting in the contest of providing redundancy, or for implementing certain bus structures. By using different wavelengths for transmitters on the PCBs and gratings to filter out a particular wavelength, a microchannel interconnect structure can be made as shown in Figure 88. This is not possible with electronic backplanes at Gbit/s rates, and it would allow new architectural approaches.

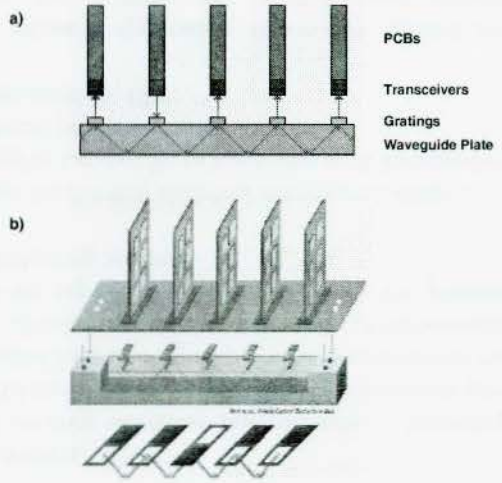


Figure 88: Microchannel interconnect (a- schematic, b- implementation)

The free-space based approaches can be classified by how an array of channels is imaged from object (laser) to image (detector). All channels of an array can be imaged through one single aperture (macro-optics), or clusters of few channels can be imaged through several smaller apertures (mini-optics), or each channel can be imaged through a dedicated aperture (micro-optics), plus all kind of combinations between these three types (hybrid optics).

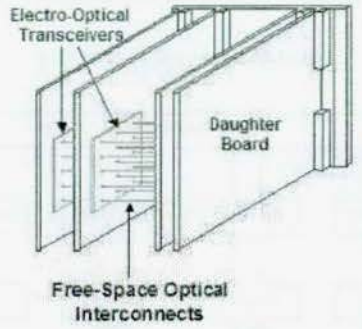


Figure 89: An example of free-space optical interconnects

One of the potential advantages of free-space optics is the low attenuation if air is used as a propagation medium. There are still losses due to reflections, but these can be minimized using anti-reflection coatings. Another potential advantage of free-space optics is the promise of increased density by using real 2D arrays.

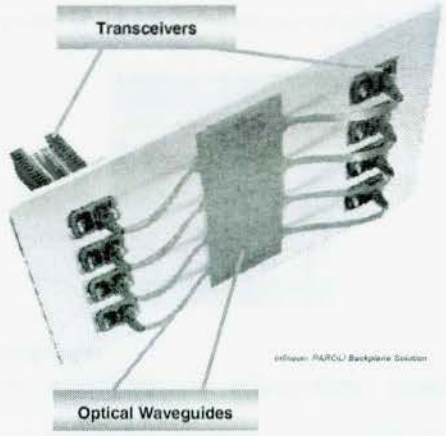


Figure 90: An example – PAROLI optical backplane

Finally, free-space interconnects are potentially less invasive to board manufacturing process and might therefore allow a shorter time to product.

A key limitation of free-space approaches is the alignment, which has to be maintained over the whole optical path and not only at the coupling interfaces, as for guided approaches. Prototype free-space interconnects have therefore been limited in length from centimetres to few tens of centimetres.

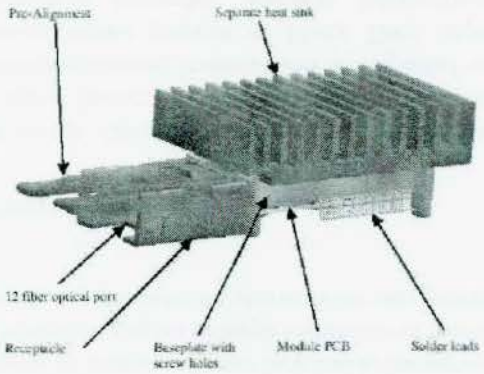


Figure 91: PAROLI transceiver module

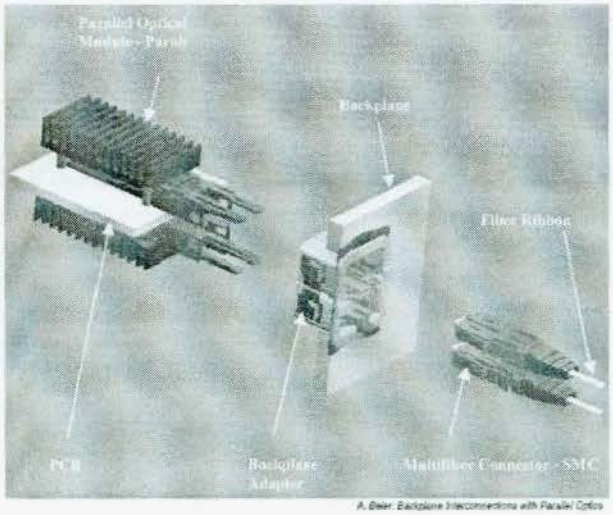


Figure 92: PAROLI backplane connection

Chapter 2

TECHNOLOGY SPECIFIC SYSTEMS

2.1 Introduction

This chapter gives a briefly description of the most important wired networking systems as well as a deeper look inside the network elements for these systems. Several examples of implementation of modules, component and functions needed in the network elements are also provided.

2.2 SONET/SDH Systems

SONET (Synchronous Optical Network - ANSI Standard) and SDH (Synchronous Digital Hierarchy - ITU Standard) are developed for circuit-switched (connection-oriented) transmission of Digitized voice and packet data over physical media (copper or fiber). They implement globally common multiplexing hierarchy, seamless inter-networking between national providers, so that they are multi-vendor compatible.

2.2.1 SONET/SDH Basics

Mostly SONET/SDH networks are implemented in a ring topology and are capable to transport different tributary signals and other transport technologies (ATM, Ethernet, and IP) as shown in Figure 93. The

specifications of SONET and SDH cover only physical layer description (OSI layer 1). These specifications (SONET and SDH) are very similar to each other in principle, and therefore, in the following text we will concentrate only on description of SDH.

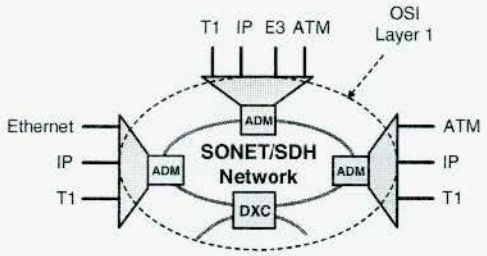


Figure 93: SONET/SDH networks

Figure 94 is a schematic diagram of a SDH ring structure with various tributaries. The mixture of different applications is typical of the data transported by SDH. Synchronous networks must be able to transmit plesiochronous signals and at the same time be capable of handling future services such as ATM. All this requires the use of various different network elements. Current SDH networks are basically made up from four different types of network element (see Figure 95). The topology (i.e. ring or mesh structure) is governed by the requirements of the network provider.

Regenerators
Regenerators, as the name implies, have the job of regenerating the clock and amplitude relationships of the incoming data signals that have been attenuated and distorted by dispersion. They derive their clock signals from the incoming data stream. Messages are received by extracting various 64 kbit/s channels (e.g. service channels E1, F1) in the RSOH (regenerator section overhead). Messages can also be output using these channels.

Terminal Multiplexers
Terminal multiplexers are used to combine plesiochronous and synchronous input signals into higher bit rate STM-N signals.

Add-Drop Multiplexers (ADM)

Plesiochronous and lower bit rate synchronous signals can be extracted from or inserted into high speed SDH bit streams by means of ADMs. This feature makes it possible to set up ring structures, which have the advantage that automatic back-up path switching is possible using elements in the ring in the event of a fault.

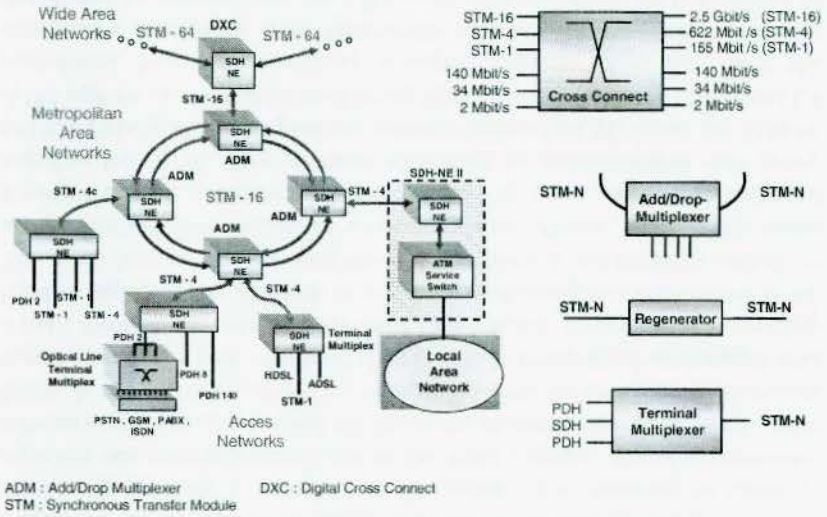


Figure 94: SONET/SDH network elements

Digital Cross-Connects (DXC)

This network element has the widest range of functions. It allows mapping of plesiochronous digital hierarchy (PDH) tributary signals into virtual containers as well as switching of various containers up to and including VC-4.

The regenerator section is the path between regenerators. Part of the overhead (RSOH, regenerator section overhead) is available for the signaling required within this layer. The remainder of the overhead (MSOH, multiplex section overhead) is used for the needs of the multiplex section. The multiplex section covers the part of the SDH link between multiplexers. The carriers (VC, virtual containers) are available as payload at the two ends of this section.

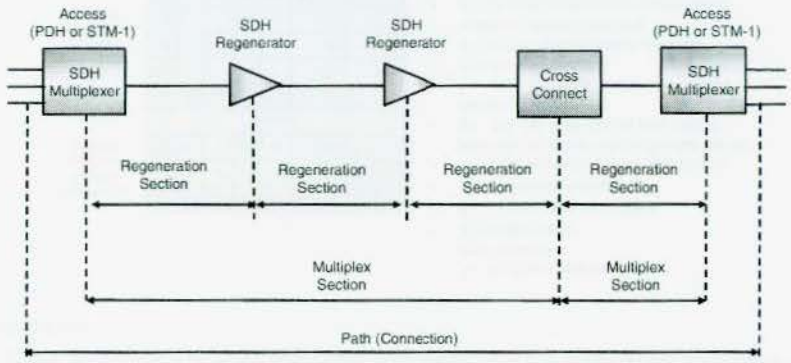


Figure 95: SDH transmission path

A frame with a bit rate of 155.52 Mbit/s is defined in ITU-T Recommendation G.707. This frame is called the synchronous transport module (STM). Since the frame is the first level of the synchronous digital hierarchy, it is known as STM-1. Figure 96 shows the format of this frame.

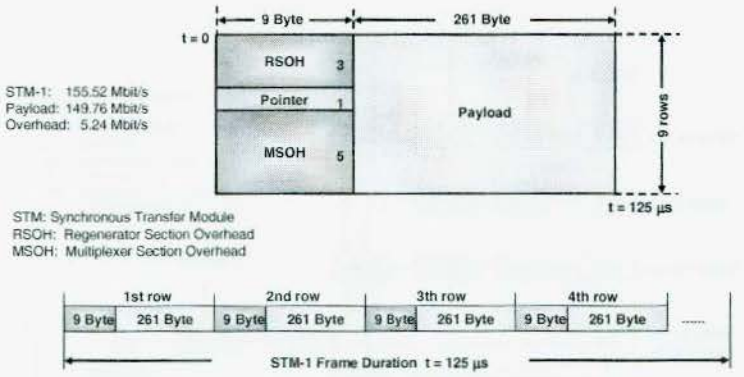


Figure 96: Format of SDH frame

A SDH frame is made up from a byte matrix of 9 rows and 270 columns. Transmission is row by row, starting with the byte in the upper left corner and ending with the byte in the lower right corner. The frame repetition rate is 125 µs. Each byte in the payload represents a 64 kbit/s channel. The STM-1 frame is capable of transporting any PDH tributary signal (140 Mbit/s).

Pointer is used to localize individual virtual containers in the payload of the synchronous transport module. The pointer may directly indicate a single VC-n virtual container from the upper level of the STM-1 frame.

Transport of tributary signals over SDH networks

The heterogeneous nature of modern network structures has made it necessary that all PDH and ATM signals are transported over the SDH network. The process of matching the signals to the network is called mapping. The container is the basic package unit for tributary channels. A special container (C-n) is provided for each PDH tributary signal. These containers are always much larger than the payload to be transported. The remaining capacity is used partly for justification (stuffing) in order to equalize out timing inaccuracies in the PDH signals. Where synchronous tributaries are mapped, fixed fill bytes are inserted instead of justification bytes. A virtual container (VC-n) is made up from the container thus formed together with the path overhead (POH). This is transmitted unchanged over a path through the network. The next step towards formation of a complete STM-N signal is the addition of a pointer indicating the start of the POH. The unit formed by the pointer and the virtual container is called an administrative unit (AU-n) or a tributary unit (TU-n). Several TUs taken together form a tributary unit group (TUG-n); these are in turn collected together into a VC. One or more AUs form an administrative unit group (AUG). Finally, the AUG plus the section overhead (SOH) forms the STM-N. ATM signals can be transported in the SDH network in C11, C12, C3 and C4 containers. Since the container transport capacity does not meet the continually increasing ATM bandwidth requirement, methods have been developed for transmitting the ATM payload in a multiple (n) C-4 (virtual or contiguous concatenation). As an example, a quadruple C-4 can be transmitted in a STM-4. Figure 97 is a summary of the mappings that are currently possible according to ITU-T Recommendation G.707 and the ATM mapping recommendations. Of interest in this context is the so-called sub-STM or STM-0 signal. This interface is used in SDH/SONET links and in radio link and satellite connections. The STM-0 bit rate is 51.84 Mbit/s.

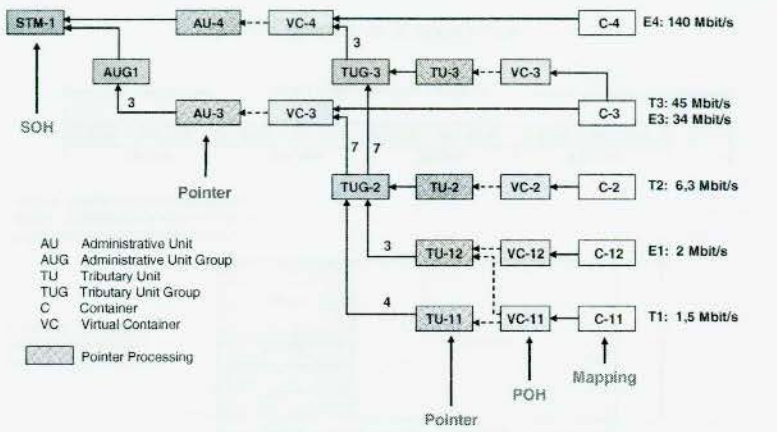
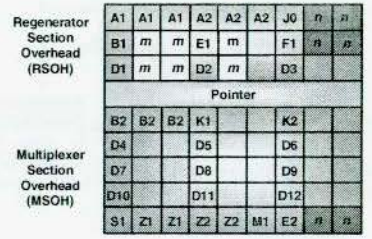


Figure 97: Mapping in SDH

Section overhead (SOH)

The first 9 bytes in each of the 9 rows are called the overhead. G.707 makes a distinction between the regenerator section overhead (RSOH) and the multiplex section overhead (MSOH). The reason for this is to be able to couple the functions of certain overhead bytes to the network architecture. Figure 98 shows the SOH structure and describes the individual functions of the bytes.



- A1, A2: Frame alignment and synchronisation
- J0: Section trace
- B1: Section parity code BIP-8
- B2: Bit interleaved parity
- F1 - User channel: used by operators to remotely recognise various physical alarms
- D1 - D3: 192 kbit/s OA&M data (RSOH network management)
- D4 - D12: 576 kbit/s OA&M data (MSOH network management)
- E1, E2: 64 kbit/s orderwire channels multiplex section monitoring
- F1: 64 kbit/s user channel (maintenance)
- K1, K2: Request/answer automatic protection switching (APS) channels
- S1: Clock quality
- M1: Error indication
- Z1, Z2: Reserve
- *m*: Media-dependent Bytes
- *n*: Bytes reserved for national use

Figure 98: Regenerator and multiplexer section overheads (SOHs)

Path overhead

The path overhead (POH) plus a container forms a virtual container. The POH has the task of monitoring quality and indicating the type of container. The format and size of the POH depends on the container type. Figure shows the POH of a VC-4 and describes functions of their bytes.

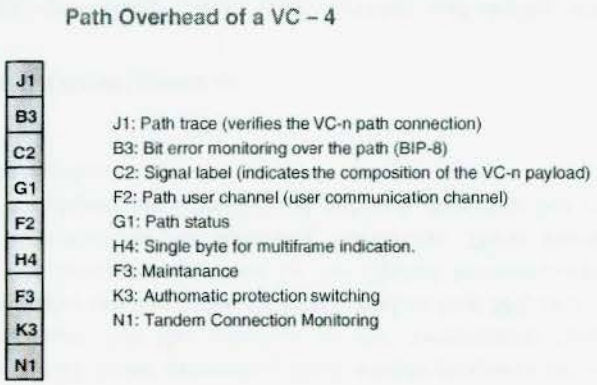
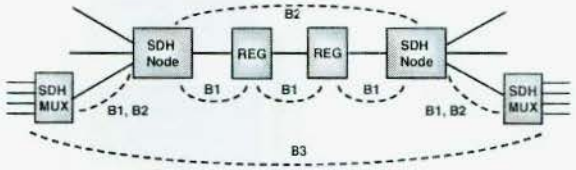


Figure 99: Path overhead (POH)

Error monitoring is implemented for each SDH section. For example, B1 byte contains BIP-8 of all bits in the previous frame using even parity, before scrambling. This is the error monitoring byte or the regenerator section.



Byte	Frame Section	Length	Monitoring Section
B1	RSOH	BIP-8	STM-1 (2430 Bytes)
B2	MSOH	BIP-24	STM-1 without RSOH
B3	POH VC-3/4	BIP-8	VC-3/4

BIP: Bit-Interleaved Parity

Figure 100: Bit error monitoring in SDH

For the multiplex section (also called line section) B2 byte contains BIP-24 calculated over all bits of the line overhead of the previous frame with even parity. In order to implement error monitoring for the whole connection, the path error monitoring byte, B3, is included in the path overhead (POF). It contains a BIP-8 over all bits of the payload of the previous frame, using even parity before scrambling.

A higher hierarchy (higher transmission rate) can be achieved by byte-wise multiplexing of four frames of the lover hierarchy as shown in Figure 101. In this way a STM-4 frame can be built from four STM-1 frames. The resulting frame has 4×270 Bytes in a column, while the frame duration remains the same (125 µs).

The main parts of a SONET/SDH interface card are shown in Figure 102. There are a number of interfaces to layer 2 processing device, which can be for example an ATM or an Ethernet or another device. Some of these interfaces are standardized and others are proprietary. The payload is first processed in the SONET/SDH interface card. This means that packets or cells from layer 2 has to be prepared for transmission and mapped to the SDH virtual container.

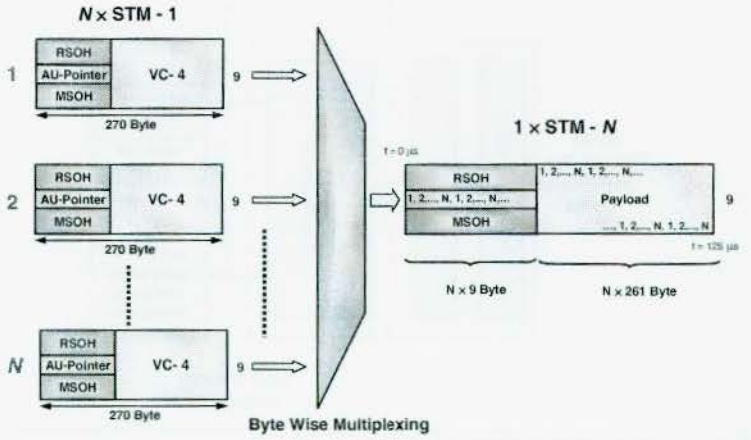


Figure 101: Byte wise multiplexing of N STM signals into one STM-N signal

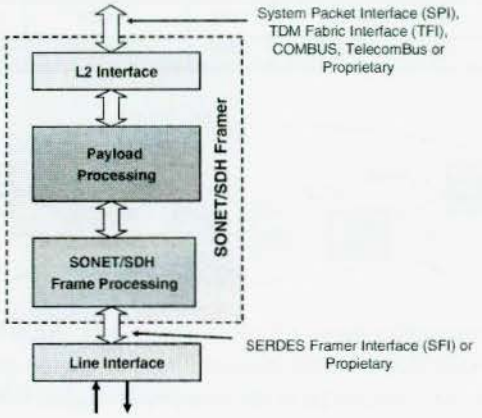


Figure 102: SONET/SDH interfaces

The SONET/SDH frame processing block inserts processes or removes the section overhead. For the interface to the transmission medium (line interface) we also can use a standardized interface (e.g. SFI) or a proprietary interface. The interfaces specified by the Optical Internetworking Forum (OIF) are described in the following subsection. These interfaces were primarily developed for SONET/SDH network elements, but can also be used in other systems.

2.2.2 SONET/SDH Interfaces

Optical Internetworking Forum (OIF) develop and deploy interoperable data switching and routing products and services that use optical networking technologies. These interfaces, the System Packet Interface (SPI) the SERDES Framer Interface (SFI), and the TDM Fabric Interface (TFI), are depicted in the following reference model:

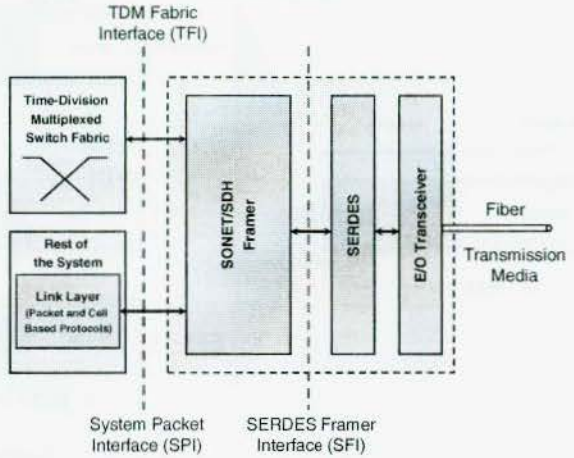


Figure 103: Electrical Interfaces developed by the Optical Internetworking Forum (OIF)

The System Packet Interface (SPI) is between the Physical Layer (PHY) device(s) and the rest of the SONET/SDH System (i.e. between the SONET/SDH Framer and the Link Layer). This interface separates the synchronous PHY layer from the asynchronous packet-based processing performed by the higher layers. As such, the SPI supports transmit and receive data transfers at clock rates independent of the actual line bit rate. It is designed for the efficient transfer of both variable-sized packet and fixed-sized cell data.

The SERDES Framer Interface (SFI) defines an electrical interface between a SONET/SDH Framer and the high speed Parallel-to-Serial/ Serial-to-Parallel (SERDES) logic. This permits the SERDES and Framer to be implemented in different speed technologies, allowing a cost- effective multiple chip solution for the SONET/SDH PHY.

The TDM Fabric Interface (TFI) defines the backplane interface between a SONET based Time division Multiplexed (TDM) framer and switch fabric. Traffic between the framer and the fabric is modelled after a SONET/SDH frame, and operates at the STS-48/STM- 16 equivalent bit rate.

The OIF has defined several different versions of these electrical interfaces:

• SPI-3	OC-48/STM-16 and below (2.48832 Gbit/s range)
• SPI-4.1 (SPI-4.2)/SFI-4.1 (SPI-4.2)	OC-192/STM-64 (10 Gbit/s range)
• SPI-5/SFI-5	OC-768/STM-256 (40 Gbit/s range)
• TFI-5	OC-48/STM-16 to OC-768/STM-256 (2.5 –40 Gbit/s range)
• SxI/CEI	Common Electrical Interfaces

2.2.2.1 *System Packet Interface (SPI)*

SPI is an electrical interface defined by the OIF to support Packet over SONET/SDH (POS). It defines an interface for efficient packet transfer between a physical layer (PHY) device and a link layer device.

SPI-3

SPI-3 is the first electrical interface defined by the OIF, which was designed to support Packet over SONET/SDH (POS) in the OC-48 (2.488 Gbit/s) and below environment. Since the SPI-3 supports data transfers at clock rates independent of the actual line bit rate, FIFOs are specified to allow the rate decoupling. Each physical layer device is required to support a minimum FIFO size of 256-bytes.

SPI-3 Interface Attributes:

- Point-to-point connection (i.e., between single PHY and single Link Layer device)
- Variable length packets
- Defines both byte-level and packet-level transfer modes
- Transmit / Receive Data Path:
  - 8 or 32 bits wide
  - Parity on the Data bus
  - Maximum clock rate of 104 MHz
- In-band PHY port selection
  - PHY port address is inserted in-band with the packet data being transferred on the data bus

  - 256 ports supported (8-bit address)
- Discrete transmit/receive control signals for Start of Packet (SOP), End of Packet (EOP), Start of Transfer, Error indications, etc.
- Transmit FIFO Status
  - Transmit Flow Control provided using either a polling or a direct status indication scheme
  - Discrete address polling control and FIFO status signals used
- FIFO buffer 256-bytes deep

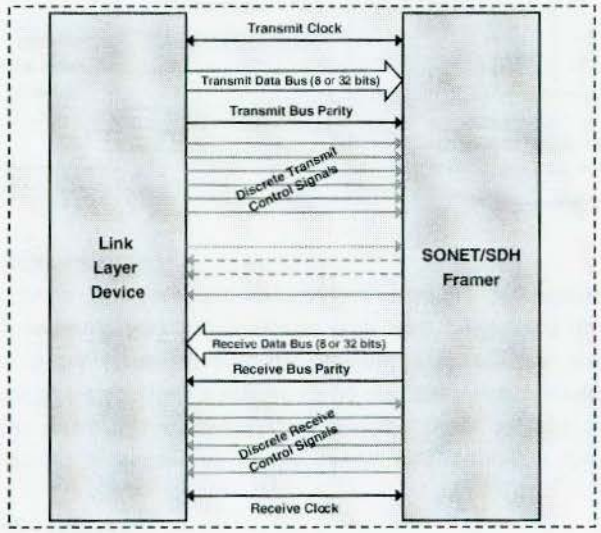


Figure 104: SPI-3 Interface

SPI-4 Phase 1 (SPI-4.1)

The SPI-4 Phase 1 interface is a packet and cell transfer interface that supports transfers at a nominal rate of 10 Gbit/s (OC-192-based) and a maximum rate of 12.8 Gbit/s. Unlike SPI-3, flow control information is passed back to the sending device on a continuous basis. The flow control information specifies whether the receiving device is full or not.

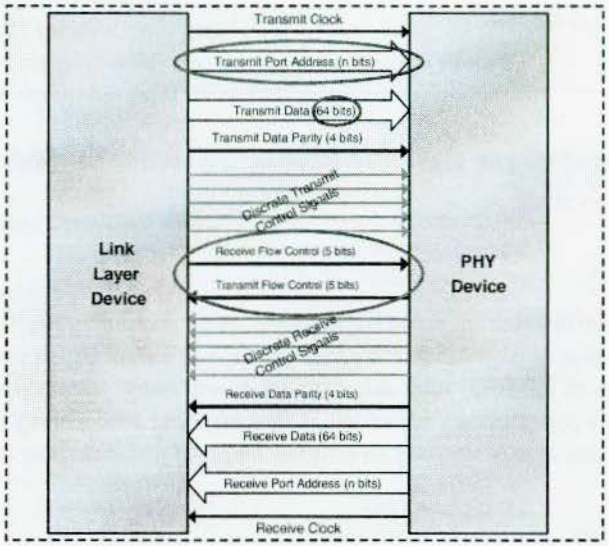


Figure 105: SPI-4.1 Interface

SPI-4 Phase 1 Interface Attributes:

- Point-to-point connection (i.e., between single PHY and single Link Layer device)
- Variable-length packets and fixed-sized cells
- Transmit / Receive Data Path:
  - 64 bits wide (optional 16 bit wide data path)
  - Source-synchronous clocking at 200 MHz
  - Parity on the Data bus
  - Single-ended HSTL Class 1 I/O
- Out-of-band port address
- Discrete address bus
- Maximum number of port address bits is not specified
- Discrete transmit/receive control signals for Start of Packet (SOP), End of Packet (EOP), Error indications, etc.
- Transmit/Receive FIFO Status interface:
  - 4-bit parallel FIFO status bus
  - Out-of-band Start-of-FIFO-Status signal
  - Flow control information is passed back to the sending device on a

continuous basis. The flow control information specifies whether the receiving device FIFO is full or not.
  - No polling
  - Transmit status bus synchronized with Transmit Clock, Receive status bus synchronized with Receive Clock
  - Same clock rate as the data bus

SPI-4 Phase 2 (SPI-4.2)

SPI-4.2 runs at a minimum of 10 Gbit/s and supports the aggregate bandwidths required of ATM and Packet over SONET/SDH (POS) applications. SPI-4 specifies a higher-speed and narrower interface than that defined in SPI-4 Phase 1 and its transmit and receive interfaces are completely separate and independent. It is well positioned as a versatile general-purpose interface for exchanging packets anywhere within a communications system.

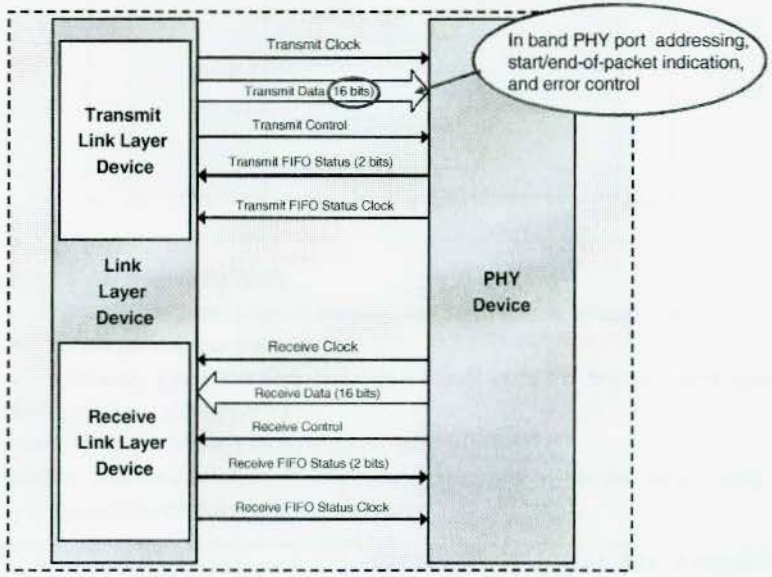


Figure 106: SPI 4.2 Interface

To be able to support both bidirectional and unidirectional connections, the transmitting and the receiving interface are designed to be completely independent of each other. This separation is mainly achieved by implementing one out-of-band status channel per direction which realises the flow control mechanism. Further the data and status channels of each interface have separate clocks (driven by the sender) so they can operate independent of the corresponding data channel. Data is always transferred in bursts which have a determined maximum length and always contain a multiple of eight data words,

The status channel is used for reporting per port remote FIFO status information. The transmitted FIFO fill level is then used to calculate the credits for the interface scheduling mechanism controlling the per port transmission data rate. Implementation can be done using either LVTTL (up to 1/4 data clock rate) or LVDS I/O (higher bandwidth operation) depending on the required update rate for the fill level information. Both the data and the status channel use an n-bit diagonal interleaved parity (DIP-n) to allow the detection of transmission errors. A four-bit parity (n = 4) is calculated over the sent data and the subsequent control word on the data channel. On the status channel a two-bit code (n = 2) which is calculated over one complete status transmission cycle is used to detect transmission errors. Depending on the number of consecutive errors the protocol core decides whether synchronisation could be established and payload transmission is possible.

The DIP-4 code covers the control word and any of the data sent after the prior control word; it is an odd-parity code that is computed "diagonally" across the 16 data lanes. Diagonal parity does not offer better performance over conventional parity in the presence of random errors. However, it allows errors isolated in one data lane (resulting for example, from a marginal PC board trace or defective I/O) to be spread across more than one parity bit, thereby increasing the likelihood of detecting those errors.

SPI-4 Phase 2 Interface Attributes:

- Point-to-point connection (i.e., between single PHY and single Link Layer device)
- Variable-length packets and fixed-sized cells
- Transmit / Receive Data Path:

  - ≤6 bits wide
  - Source-synchronous double-edge clocking with a 311 MHz minimum
  - 622 Mbps minimum data rate per line
  - LVDS I/O (IEEE 1596.3 – 1996, ANSI/TIA/EIA-644-1995)
  - Control word extension supported
- In-band PHY port addressing
- Support for 256 ports (suitable for STS-1 granularity in SONET/SDHapplications (192 ports) and Fast Ethernet granularity in Ethernet applications (100 ports))
- Extended addressing supported for highly channelized applications
- In-band start/end-of-packet indication, error-control code
- Transmit/Receive FIFO Status interface:
  - 2-bit parallel FIFO status bus
  - In-band Start-of-FIFO-Status signal
  - Source-synchronous clocking
  - Flow control information is passed back to the sending device on a continuous basis. The flow control information specifies whether the receiving device FIFO is full or not.
  - No polling
  - Uses either CMOS LVTTL I/O (3.3V) or LVDS I/O (IEEE 1596.3 – 1996, ANSI/TIA/EIA-644-1995)
  - Run at a maximum of ¼ data path clock rate for LVTTL I/O or at full datapath clock rate for LVDS I/O
- Segmentation of data is in multiples of 16 bytes with the exception of transfers that terminate with an End-of-Packet indication.

SPI-5

SPI-5 runs at a minimum of 40 Gbit/s and supports the aggregate bandwidths required of ATM and Packet over SONET/SDH (POS) applications. SPI-5 specifies a 16-lane interface with transmit and receive interfaces completely separate and independent. Each direction of the SPI-5 bus has its own Pool Status Channel that is sent separately from the corresponding data path. This makes it possible to decouple the Transmit and Receive interfaces making SPI-5 suitable for both bidirectional and unidirectional link layer devices.

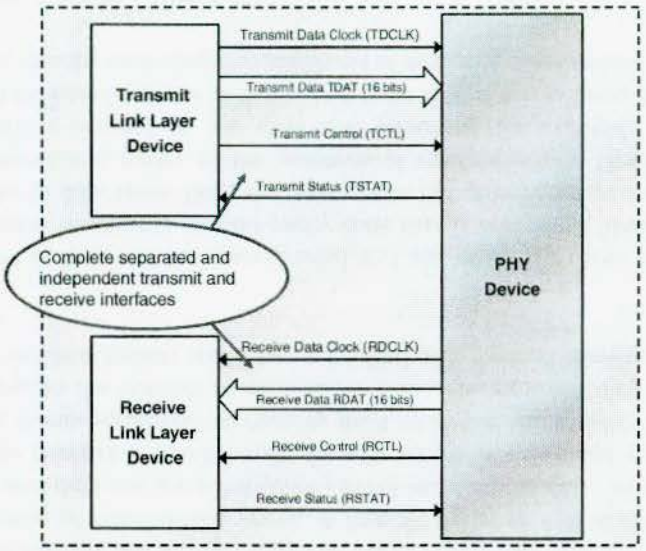


Figure 107: SPI-5 Interface

On the data path, the interface signals consist of a 16 parallel data lanes (TDAT , RDAT) together with a clock (TDCLK, RDCLK) and a control signal (TCTL, RCTL). In order to control transition density, the control and data signals are independently scrambled by an X11 + X9 + 1 linear feedback shift register (LFSR) stream cipher.
The clock is source-synchronous (originating together with the data at the source end) and runs at one quarter the baud rate of the data. In-band signaling is used on the data path, in which control signals may be inserted between data transfers. The control line is used to distinguish between periods of data and control information; it is low during data transfer and high otherwise. In-band control information includes the means for indicating start-of-packet (SOP), end-of-packet (EOP), the destination port address of the transfer, and an error-detection code.

The SPI-5 interface can be operated at data rates from 2.5 to 3.125 Gbit/s to transport various payloads, including ATM, packet over SONET/SDH, (POS) and Ethernet frames.

Flow control information is sent out-of-band in a serial status line (TSTAT, RSTAT) that runs at the same bit rate as the data path. These status lines are also scrambled with the same scheme employed in the data path.

Out-of-band flow control has the advantage of keeping transmit and receive interfaces independent of each other. If in-band flow control were used, then the flow control for one data path would be carried in the data path in the other direction. This would be of no major consequence if both transmit and receive functions could be integrated in one link layer device.

With the present state of IC technology, however, it is not possible to achieve any meaningful integration of both functions into one device. Hence, most link layer implementations will consist of separate, unidirectional devices. Consequently, an in-band flow control scheme would have required yet another interface between those two devices to relay flow control information onto the opposite data path. So while it might appear "cleaner" to have all control information sent in band, practical realities demand that flow control be sent out-of-band.

SPI-5 Interface Attributes:

- Point-to-point connection (i.e., between single PHY and single Link Layer device)
- Variable-length packets and fixed-sized cells
- Transmit / Receive Data Path:
  - 16 bits wide
  - Source-synchronous clock at one quarter the data rate
  - 3.125 Gbit/s maximum, 2.488 min. data rate per line
  - CML I/O per the SxI-5 common electrical specification
- In-band PHY port addressing
- Support for 256 ports
- Extended addressing (up to 2^144 ports) supported for highly channelized applications
- Control words carry In-band port address, start/end-of-packet indication and error-control code
- Transmit/Receive Pool Status channel:
  - Operates at the same clock rate as the data path
  - Bit Pool status indication
  - In-band Pool Status framing

- Electrical, training and scrambling functions common with data path
- In-band training

- Segmentation of data is in multiples of 16 words (32 bytes) for nominal burst transfer. EOP packets can be shorter (allowing data burst lengths to be sized appropriate to the application).

2.2.2.2 *SERDES Framer Interface (SFI)*

SFI-4 Phase 1

The SFI-4 phase 1 interface supports transmit and receive data transfers at clock rates locked to the actual line bit rate. It is optimized for the pure transfer of data. There is no protocol or framing overhead.

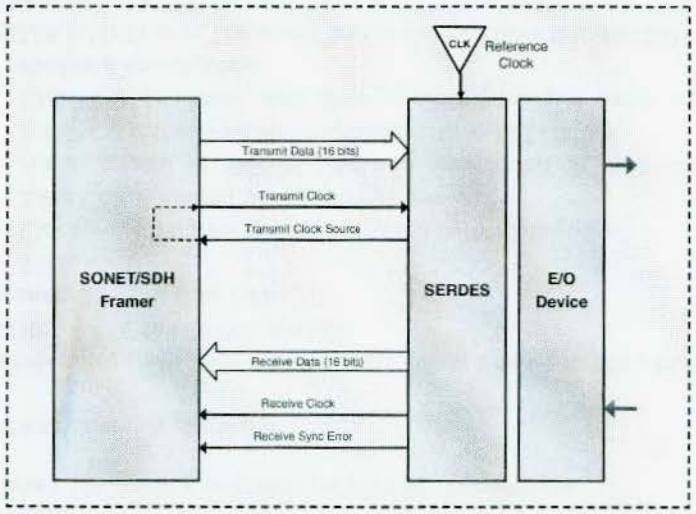


Figure 108: SFI-4.1 Interface

SFI-4 Phase 1 Interface Attributes:

- Point-to-point connection (i.e., between single Framer and single SERDES device)
- Transmit / Receive Data Path:

  - 16 bits wide
  - Source-synchronous clocking at 622.08 MHz (311.04 MHz with Double Data Rate sample optional)
  - 622 Mbit/s minimum data rate per line
  - An aggregate of 9953.28 Mbit/s is transferred in each direction
  - Timing specifications allow operation up to 10.66 Gbit/s
  - LVDS I/O (IEEE Std 1596.3-1996)
- The 622.08 MHz Framer Transmit Clock sourced from the SERDES
- Uses a 622.08 MHz LV-PECL reference clock input
- A low-speed (LVTTL) Receive Loss of Synchronization Error is signaled when the Receive Clock and Receive Data are not derived from the received optical signal

SFI-4 Phase 2

The SFI-4 phase 2 interface supports a narrower 4 bit transmit and receive data path. In addition to the Framer to SERDES interface, SFI-4.2 can be used to transmit higher rate Forward error correction transfers between the FEC and SERDES.

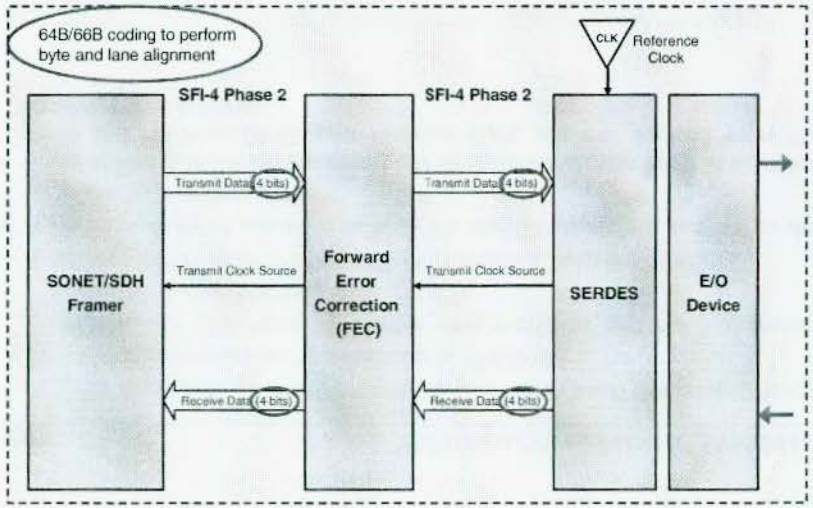


Figure 109: SFI-4.2 Interface

The interface uses 64B66B coding to perform byte and lane alignment functions and encode the clock. The interface is protocol agnostic, supporting 10G Ethernet, 10G Fibre Channel, OC-192/STM-64, G.709, proprietary FEC coding and other proprietary data streams.

SFI-4 Phase 2 Interface Attributes:

- Point-to-point connection (i.e., between single Framer to FEC, Framer to SERDES or FEC to SERDES device)
- Transmit / Receive Data Path:
  - 4 bits wide
  - Clockless interface (clock is embedded in the data path)
  - 2.488 Gbit/s minimum data rate per line
  - An aggregate of 9953.28 Mbit/s is transferred in each direction (timing specifications allow operation up to 12.5 Gbit/s)
  - CML I/O (electrical and jitter specifications per SxI-5 common electrical specification
- A 622.08 MHz Framer Transmit Clock is sourced from the SERDES
- A low-speed (LVTTL) Receive Loss of Synchronization Error is signaled when the Receive Clock and Receive Data are not derived from the received optical signal

SFI-5

The SFI-5 interface supports a 16 bit transmit and receive data path. In addition to the Framer to SERDES interface, SFI-5 can be used to transmit higher rate forward error correction transfers between the FEC and SERDES. The interface assumes scrambled data maintain DC balance. Clock and lane de-skew are accomplished using additional signals. The interface supports OC-768/STM-256 and FEC coded OC-768 data.

SFI-5 Interface Attributes:

- Point-to-point connection (i.e., between single Framer to FEC, Framer to SERDES or FEC to SERDES device)
- Transmit / Receive Data Path:
  - 16 bits wide
  - 622 MHz clock
  - 2.488 Gbit/s minimum data rate per line

  - An aggregate of 39.8 Gbit/s is transferred in each direction (timing specifications allow operation up to 50 Gbit/s)
  - CML I/O (Electrical and jitter characteristics per SxI-5 common electrical specification - CEI)
- A 622.08 MHz Framer Transmit Clock is sourced from the SERDES
- A de-skew channel included in both RX and TX directions is used to de-skew the 16 lane data path.
- A low-speed (LVCMOS) Receive Loss of Synchronization Error is signaled when the Receive Clock and Receive Data are not derived from the received optical signal.

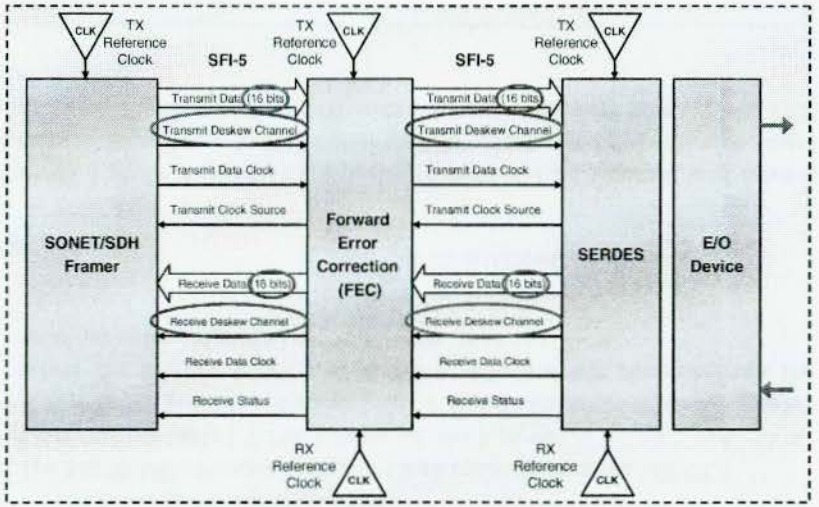


Figure 110: SFI-5 interface

2.2.2.3 Common Electrical Interface (CEI)

A faster electrical interface is required to provide higher density and/or lower cost interfaces for payloads of 10 Gigabits and higher, including SERDES to Framer Interface (SFI), System Packet Interface (SPI), TDM-Fabric to framer Interface (TFI). The Common Electrical Interface (CEI) Implementation Agreement includes:

- Electrical and jitter methodologies for new high speed interfaces and including the following older OIF interfaces: SxI-5, SFI-4.2, SFI-5.1, SPI-5.1 and TFI-5.
- A CEI-6G-SR specification for Data lane(s) that support bit rates from 4.976 to 6.375 Gsymbos/s over Printed Circuit Boards. Physical reach from 0 to 200 mm and up to 1 connector
- A CEI-6G-LR specification for Data lane(s) that support bit rates from 4.976 to 6.375 Gsymbol/s over Printed Circuit Boards. Physical reach from 0 to 1 m and up to 2 connectors.
- A CEI-11G-SR specification for: Data lane(s) that support bit rates from 9.95 to 11.1 Gsymbol/s over Printed Circuit Boards.

The Implementation Agreement defines applicable data characteristics (e.g. DC balance, transition density, and maximum run length), channel models and compliance points/parameters supporting the physical reach and conditions. The Implementation Agreement specifically excludes any pin-out, management interface, power-supply specification, connector or higher-level activity such as addressing or error control. It does not endorse or specify any particular data protocol.

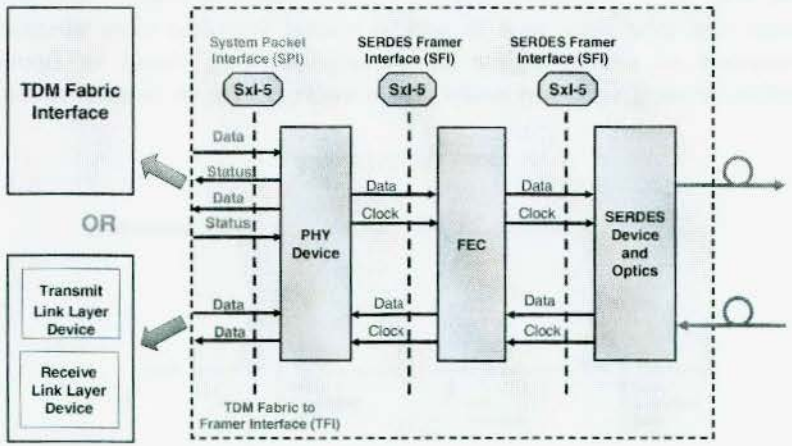


Figure 111: Common electrical interface (CEI)

The signal is a low swing differential interface. This implies that the receiver has a wide common mode range (within the max. absolute input voltages).

The signal is a low swing differential interface. This implies that the receiver has a wide common mode range (within the max. absolute input voltages). The signal paths or CEI lanes are unidirectional point-to-point connections. Each CEI lane is made up of a balanced differential pair. A CEI link can be comprised of a unidirectional single lane or parallel lanes in either transmit or receive direction. A CEI link does not imply duplex operation. See Figure 112 for more information, which shows 2 CEI Links, in the receive and transmit directions.

The data path shall allow single and multi-lane applications and support AC coupling and Hot Plug. The achieved bit error ratio of lower than 10-15 per lane has been specified. The short and long reach links should interoperate for signal path lengths up to 200 mm.

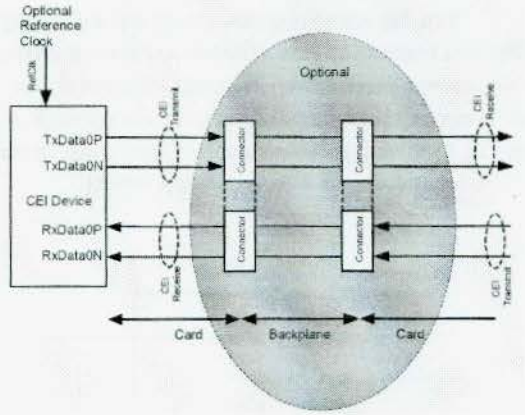


Figure 112: Signal diagram of the common electrical interface (CEI)

Cross talk can arise from coupling within the connectors, on the PCB, the package and the die. Cross talk can be categorized as either Near-End or Far-End Cross talk (NEXT and FEXT). In either of these categories, the amount of cross talk is dependent upon signal amplitudes, signal spectrum, and trace/cable length.

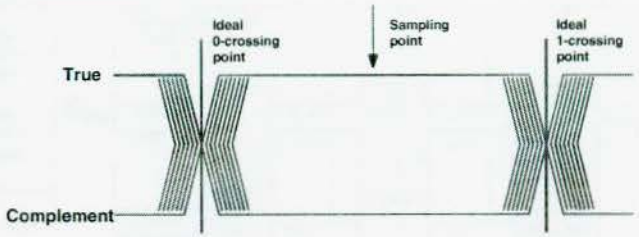


Figure 113: CEI jitter

Jitter is defined as the deviation of the signal transition from an origin, usually its mean. This deviation has an amplitude and an associated spectrum. High frequency jitter is defined by a 1st order high pass phase filter with a corner frequency equal to the ideal bandwidth of the clock and data recovery (CDR) circuit. The low frequency jitter or wander is defined by a 1st order low pass phase filter with a corner frequency equal to the bandwidth.

The receive eye mask specifies the jitter and amplitude at the receiver as shown in Figure 114. The horizontal limit specified in the eye mask represents the total jitter seen at the receiver input. The receiver must tolerate at least the receiver eye template and jitter requirements. The receive eye mask specifies the jitter at reference points B and D. Similar eye max is specified for the transmitter (points A and C)

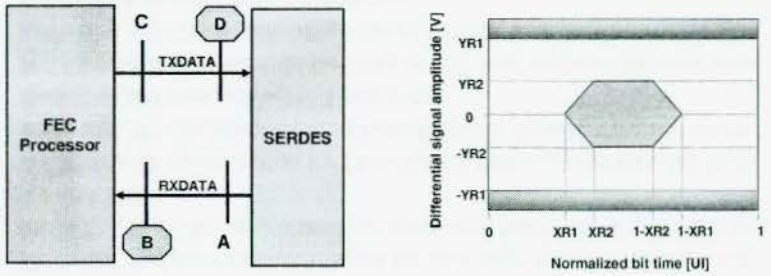


Figure 114: CEI eye mask

Figure 115 illustrates skew and relative wander. Wander is the variation in the phase of a signal (clock or data) after filtering with a low pass filter.

Skew is the constant portion of the difference in the arrival time between two signals.

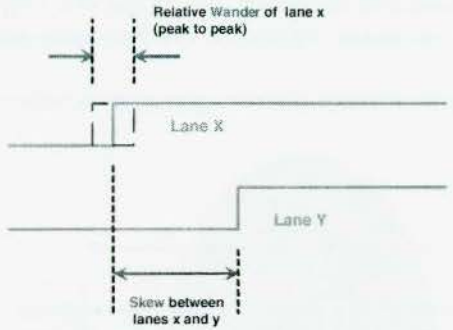


Figure 115: CEI wander and skew

While the protocol layer will control some of the lane to lane skew, the electrical level is allowed up to 500 ps of lane-to-lane skew caused by the driver circuitry and associated routing. The allowed line-to-line skew that is caused by the driver & receiver circuitry and associated routing is limited to 1000 ps (that is 500 ps for the driver and 500 ps for the Rx).

2.2.2.4 TDM Fabric Interface (TFI)

TFI-5 is a SONET/SDH-like backplane interface, either electrical or optical, connecting a framer to a TDM switch. A system reference model is shown in Figure 116. A variety of framers, including SONET/SDH, optical transport networks (OTN), 10 GE WAN PHY and 10 GE LAN PHY framers, are shown to operate simultaneously in the system.

As the services provided by the TFI-5 link in the Framer-to-Fabric direction, are identical to that provided in Fabric-to-Framer direction, there is only one definition of a TFI-5 link. It is applicable for either direction of traffic. Separate receive and transmit link definitions are unnecessary. All instantiations of the TFI-5 links in Figure 116 are identical: a 2.488 Gbit/s (or 3.1104 Gbit/s) link locked to a common system reference clock.

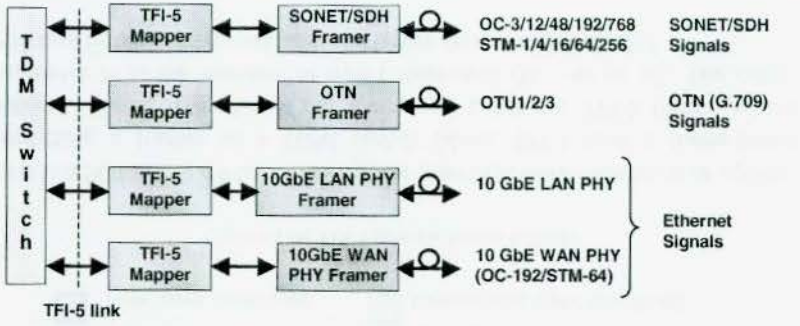


Figure 116: Reference diagram of TDM fabric interface (TFI)

The TFI-5 link has the following general characteristics:

- A TFI-5 link is a point-to-point connection between a framer and a TDM switch fabric device.
- The X7 + X6 + 1 scrambling polynomial is used to ensure rich transition density, B1 is used for link error monitoring, and B2 can be used for connection monitoring.
- TFI-5 is defined using a layered approach. Three different layers are defined: Link layer, Connection layer and Mapping layer. The Link layer is generated and terminated by the TFI-5 link source and sink devices and its extent is a TFI-5 link. The Connection layer is generated/terminated by the framer and its extent is the connection of a tributary (an STS-1 time-slot) from the Ingress framer to the Egress framer, passing through the TDM switch fabric. The Mapping layer is also generated/terminated by the framer. It provides the transport of client signals over one or more Connection layer time-slots.
- All TFI-5 links are frequency locked and all theTFI-5 frames need to be relatively frame aligned within the de-skew window of the fabric. Methods of aligning client signals to the TFI-5 frame include pointer processing and multiplexing/demultiplexing and/or through a mapping process.
- TFI-5 supports de-skew between lanes originating from multiple framers or fabric devices. De-skew algorithm must operate without additional signal lanes.
- Capable of driving at least 30 inches of PCB with 2 connectors for intra-

shelf environments and at least 100 meters over optics for intershelf environments.
- Capable of bit error rate of 10E-12 for PCB and for Optical links.

There are three layers defined for TFI-5: Link, Connection and Mapping. Figure 117 shows the hierarchical relation between the layers.

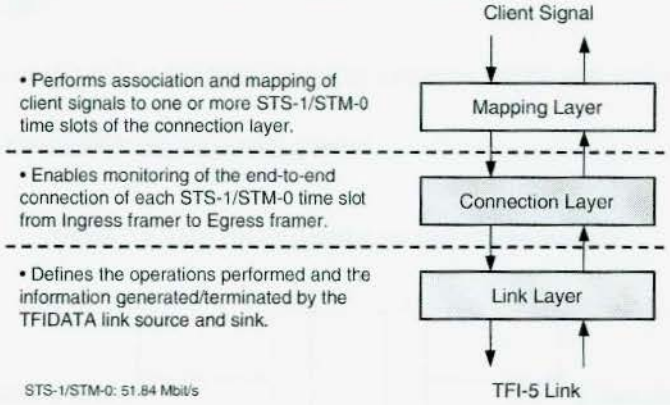


Figure 117: TFI-5 layered approach

The TFI-5 Link layer defines the operations performed and the information generated/terminated by the TFIDATA link source and sink devices. The TFI-5 Link layer is generated and terminated at each TFI-5 link.

The TFI-5 Connection layer enables monitoring of the end-to end connection of each STS-1 time-slot from Ingress framer to Egress framer, passing through the TDM switch fabric.

The TFI-5 Mapping layer performs the association and mapping of client signals to one or more STS-1 time-slots of the Connection layer. In order to carry client signals with bandwidths greater than that of a single TFI-5 link, the TFI-5 Mapping Layer can group a set of STS-1 time-slots transported in different TFI-5 links (inverse multiplexing).

There are only three signals defined in TFI-5; a data signal (TFIDATA), a reference clock signal (TFIREFCK) and an 8 kHz frame boundary reference (TFI8KREF).

Signal Name	Function
TFIDATA	The TFI-5 Data (TFIDATA) signal carries the data between the Framer and the Switch Fabric. The same signal definition is applicable to data trarsfer in the Framer to Fabric direction, and the Fabric to Framer direction.
TFIREFCK	The TFI-5 Reference Clock (TFIREFCK) signal provides timing reference to all the TFI-5 data (TFIDATA) signals in a system.
TFI8KREF	The TFI-5 8 kHz Frame Reference (TFI8KREF) signal provides reference to frame boundaries for all the devices in a TFI-5 system.

Table 4: TFI-5 signal definitions

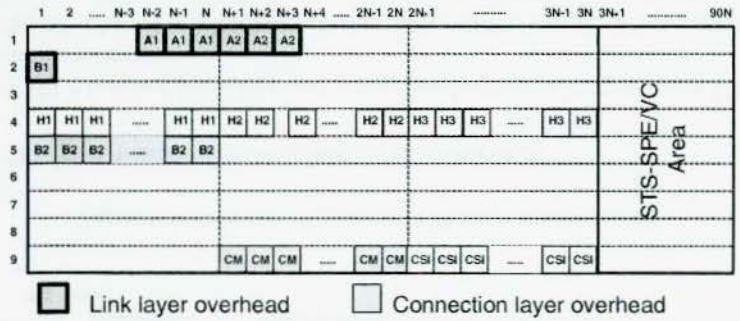


Figure 118: TFI-5 generic frame format

TFI-5 is a SONET/SDH-based backplane interface, either electrical or optical, connecting a framer to a TDM switch fabric. TFI-5 uses a frame-based protocol. Figure 118 shows the format of a generic TFI-5 frame, where parameter N is the number of STS-1 time-slots (N = 48 or 60). The colors depicting bytes correspond with the layers shown in Figure 117.

Figure 119 shows the extent of each TFI-5 layer using a simplified STS-1 cross-connect model. This simplified model separates the Ingress framers (receivers) and the Egress framers (transmitters) in order to simplify the description.

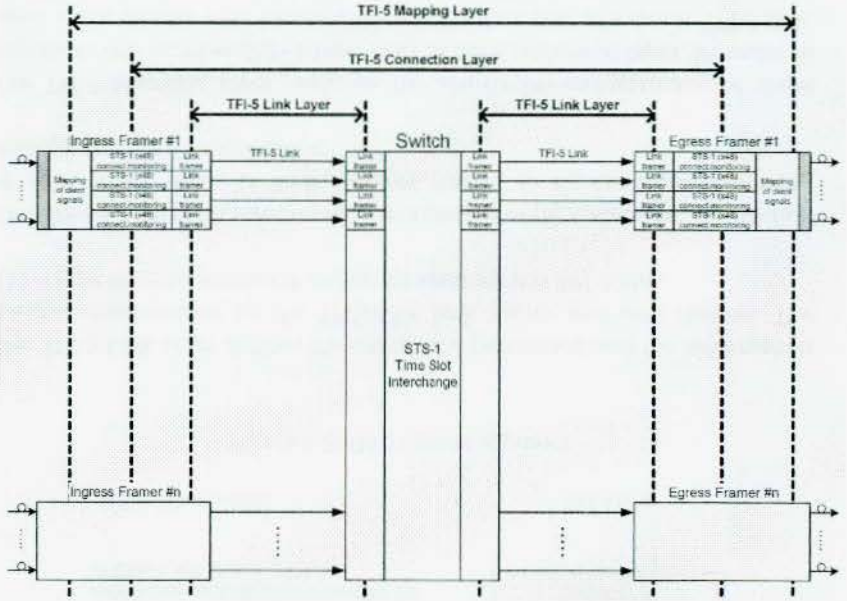


Figure 119: TFI-5 Layers Example (an STS-1 cross-connect)

2.2.3 Case Study: SONET/SDH Framer

Functional layering of a typical SONET/SDH framer device is as follows:

L2 Interface
- Connection to Datalink or MAC layer
- 8 – 256 bit wide x to yy MHz busses

Payload Processing
- Preparation of packet-oriented services:
- ATM and IP for container transport
- Cell/frame delineation, idle capacity filling

SDH Frame Processing
- Frame structuring and OAM processing

Line Interface
- Parallel-to-serial conversion and clock recovery
- Today: Integrated in the framer device up to STM-16/OC-48

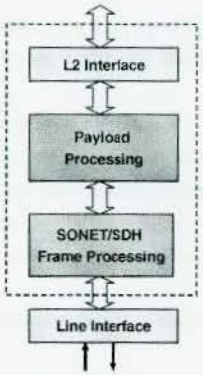


Figure 120: SONET/SDH framer

An example:

TranSwitch PHAST-12E is a modular SONET/SDH Framer. It transports ATM cell, IP packets and traditional T1/T3 traffic and scales from OC-3 (155Mbit/s) up to OC-48 (2.5 Gbit/s).

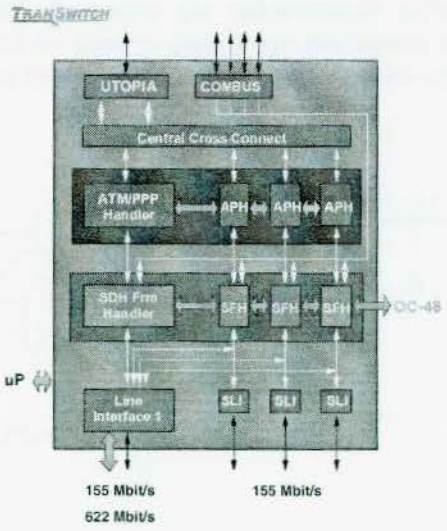


Figure 121: An example - TranSwitch PHAST-12E

This framer is fully integrated and contains WAN compliant mixed signal serial interfaces in standard CMOS. At the system level it supports add-drop multiplexing, cross-connect function, and protection switching. Technology used is 0.35 µm CMOS, 11 mm die, 560 kcells, in a 474 CBGA dual power package.

The two main functions of the line interface are clock recovery/division and serialization/deserialization (SERDES) of the data streams. SERDES and clock recovery/division logic are subject for bipolar/SiGe design. The designers try to convert their design as soon as possible to standard CMOS technology in order to make the integration with higher layer framer functions easy. Designers choose an optimal data path width (W) and clock frequency f_p for frame and payload processing. The lover frequency f_p the larger data path must be.

There is a trade-off between large data paths, which require an occupation of a large chip area and a more difficult routing within the chip, and high processing speed that requires faster and more expensive devices as well as a more accurate clock distribution and calculation of the critical timing paths.

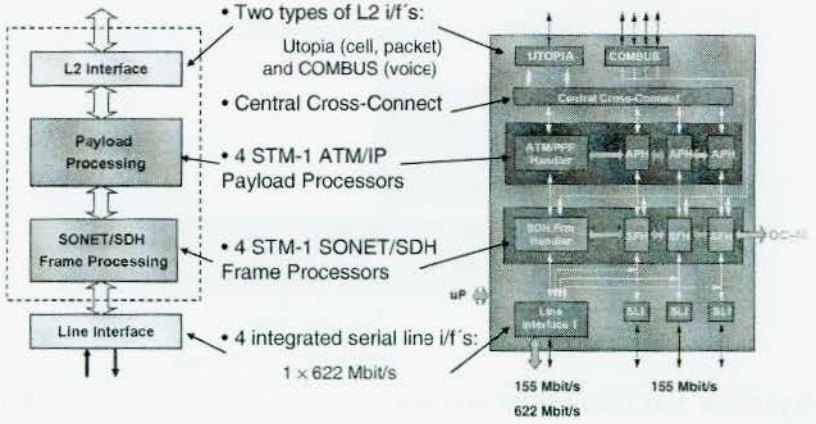


Figure 122: Framer analysis

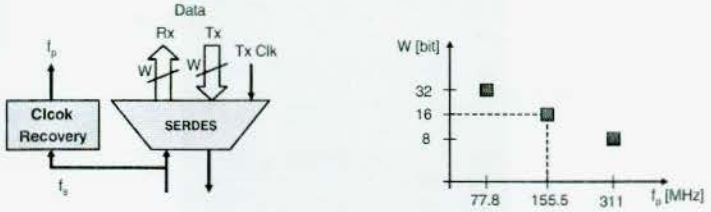


Figure 123: Line interface

Some examples of the common used widths and clock rates for frame and payload processing are listed in Table 5. The last column shows the suitable CMOS technology generation for implementing higher layer framer functions.

SDH	f_s [MHz]	W [bit]	f_p [MHz]	CMOS*
STM-1	155.5	8	19.4	≤ 0.25 µm
STM-4	622.0	8	77.8	0.18 µm
STM-16	2488.3	32	77.8	0.18 µm
STM-64	9953.3	32	311.0	0.09 – 0.11 µm
STM-256	39813.1	128	311.0	0.09 – 0.11 µm

* Suitable CMOS generation for higher layer framer functions

Table 5: Common parameters of line interfaces for different speeds

Frame and cell payload processing are mostly performed in a dataflow pipeline (in contrast to store-and-forward packet processing). Frame processing includes generation and termination of the SONET/SDH section and path overhead, frame synchronization, pointer management as well as error monitoring at these levels.

Figure 124 shows an example of payload processing where ATM cells are mapped into the virtual container of a STM signal. The operations need to be performed are:

- Calculation/checking of the header error control (HEC) field (see Section 2.4)
- Cell delineation by HEC detection
- Payload scrambling/descrambling
- Idle cell insertion/discard.

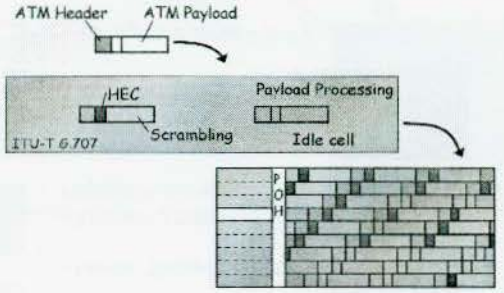
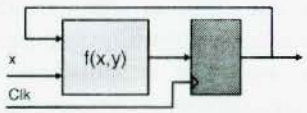


Figure 124: ATM over SONET/SDH

For the implementations of high-speed function a synchronous pipelined design is required. The functions are implemented in sequential logic by using finite state machines (FSMs). The sequential depth of such FSMs is a very important designing parameter. It is calculated as shown in Figure 125 as a sum of all delays in the longest path through the logic. It consist of all delays in the gates that are needed to implement the function f(x,y), all propagation delays through interconnection wires, set-up time of the resister as well as propagation delay through the register. The maximum sequential depth is then given by *N*max.



$$T_{clk} > \sum T_{logic} + T_{setupFF} + T_{pdFF} =$$
$$\sum (T_{gate} + T_{wire}) + T_{setupFF} + T_{pdFF}$$

$$N < \frac{T_{clk} - T_{setupFF} - T_{pdFF}}{T_{gate} + T_{wire}}$$

Figure 125: Finite state machine (FSM) sequential depth

An example:

Data path:
- Width W = 16 bit
- Data rate r = 2.5 Gbit/s
- T_{clk} = 1/155.5 MHz = 6.43 ns

CMOS datasheet:
- T_{gate} = 80 ps
- $T_{setupFF}$ = 240 ps
- T_{wire} ≈ 50 % Tgate = 40 ps
- T_{pdFF} = 80 ps

$\Rightarrow$ $N < 50.91$

That means that the maximum sequential length is $N\max = 50$. Thus, all pipeline stages should be smaller then 50. Figure 126 shows an implementation of the ATM cell processing. The required functions are shown by means of state diagrams and a pipeline scheme. It is important that all pipeline stages (logic between two registers) must be shorter than the maximum sequential deep $N\max$. If this is not the case than the particular function has to be divided into two or more parts that satisfy this requirement.

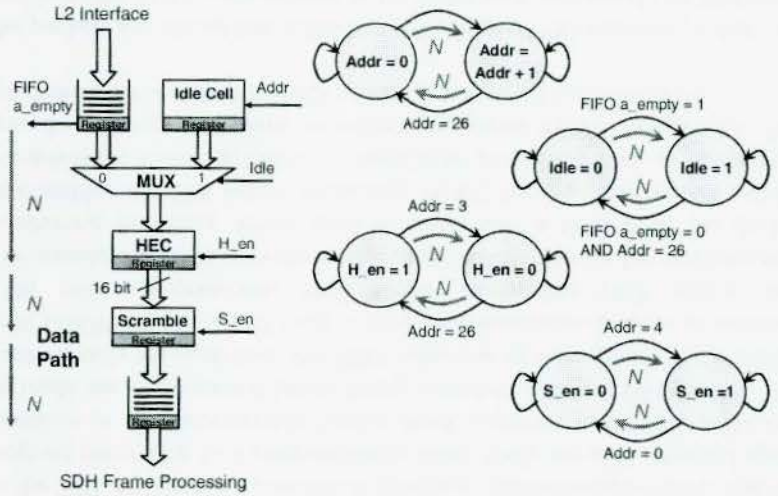


Figure 126: ATM cell processing

Some examples of framer devices are given below.

The MARS2G5 P-Pro SONET/SDH interface device provides a versatile solution for quad OC-3, quad OC-12, and for single OC-48 linear datacom/telecom applications. Constructed using COM2 CMOS modular process, this device incorporates integrated SONET/SDH framing, section/line/path termination, pointer processing, and data engine blocks. The device provides complete encapsulation and de-encapsulation for packet and ATM streams into and out of SONET/SDH payloads.

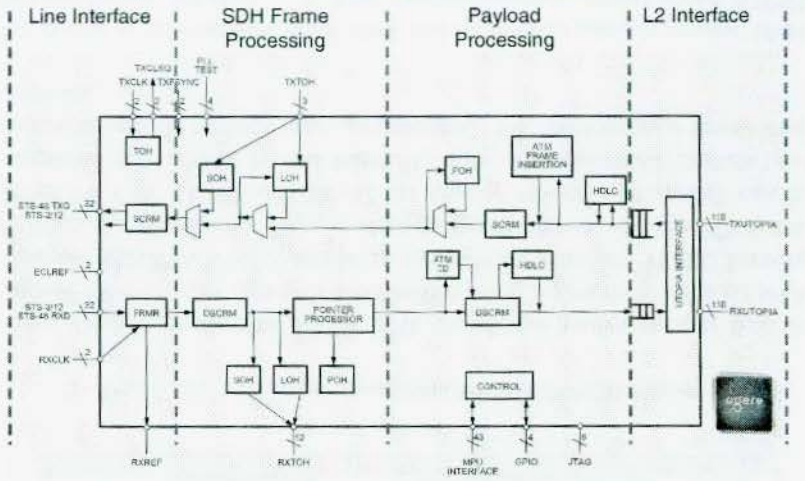


Figure 127: An example – Agere MARS2G5 P-Pro (SONET/SDH framer)

Communication with the MARS2G5 P-Pro device is accomplished through a generic microprocessor interface. The device supports separate address and data buses. With the MARS2G5 P-Pro device, support for different types of applications for OC-3/OC-12/OC-48 data equipment is possible, enabling dramatic system cost reduction and the ease of development of extremely competitive solutions.

This device integrates the SONET/SDH network termination functions with a generic cell/packet delineation circuit. The interface rates supported are STS-48/STM-16, quad STS-12/STM-4, and quad STS-3/STM-1. The UTOPIA interface can process and hand off up to 16 channels transported within an STS-N payload. The concatenation levels supported by this device are STS-1, STS-3c, STS-6c, STS-9c, STS-12c, STS-15c, . . . , STS-45c, and STS-48c. The data formats processed by this device are ATM cells or HDLC framed packets such as PPP or GFP framed packets.

This SONET line interface integrated circuit (IC) implements a four-port, sixteen-channel SONET/SDH interface for asynchronous transfer mode (ATM) and packet over SONET (POS) mappings at the STS-3 (STM-1), STS-12 (STM-4), or STS-48 (STM-12) rate. This device also supports direct cell/packet over fiber at up to 2.488 Gbit/s.

The receive path terminates and processes section, line, and path overhead. It performs framing (A1, A2) and descrambling, detects alarm conditions, and monitors section, line, and path BIP-8s (B1, B2, and B3), accumulating error counts for each level for performance-monitoring purposes. Line and path remote error indications (M1, G1) are also accumulated. The payload pointers (H1, H2) are interpreted and the synchronous payload envelope (SPE) is extracted.

When used to implement an ATM UNI, the device performs cell delineation on the SPE. HCS error correction is provided. Idle/unassigned cells may be dropped according to a programmable filter. Cells are also dropped upon detection of an uncorrectable header check sequence error. The ATM cell payloads are descrambled before being passed to a 32-cell FIFO buffer. The received cells are read from the FIFO using one of four generic 8-/16-/32-bit wide UTOPIA level 2 and level 3 compliant interfaces. Counts of received ATM cells, uncorrectable HCS errors, correctable HCS errors, and idle/unassigned ATM cells are accumulated independently for performance-monitoring purposes. When used to implement a POS UNI, the device descrambles the SPE before extracting HDLC frames. The control escape characters are removed. Descrambling can be performed after control escape byte de-stuffing (or before to control malicious HDLC expansion). The optional 16- or 32-bit error check sequence is verified for correctness.

The packets are placed into a 256-byte FIFO buffer. The received packets are read from the FIFO using generic 8-/16-/32-bit wide enhanced UTOPIA level 2 and level 3 compliant interfaces. Counts of errored/dropped packets are accumulated independently for performance-monitoring purposes. The device POS implementation also allows the optional detach of a per-channel provisionable PPP header.

The transmit path inserts section, line, and path overhead. It inserts the framing pattern (A1, A2), performs scrambling, inserts AIS (optionally), and calculates and inserts section, line, and path BIP-8s (B1, B2, B3). Line and path remote failure indications (M1, G1) are inserted based on received BIP-8 errors. The payload pointers (H1, H2) are generated, and the SPE is inserted.

When used to implement an ATM UNI, ATM cells are written into an internal four-cell (per channel) FIFO buffer using a generic 8-/16-/32-bit wide

UTOPIA level 2 and level 3 compliant interface. Idle/unassigned cells are automatically inserted when the internal FIFO is empty. The device provides generation of the header check sequence and scrambles the ATM payload. Also supports cell-based UNI per I.432 (i.e., ATM over fiber).

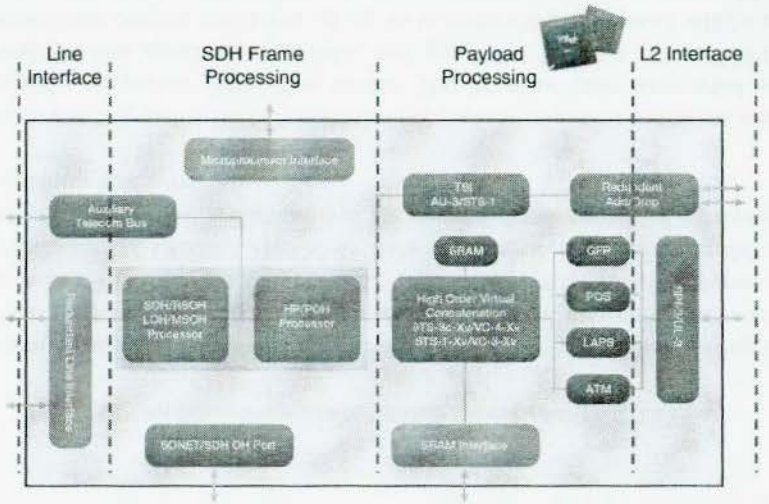


Figure 128: An example – Intel WG4500 (SONET/SDH framer)

When used to implement a POS UNI, the device writes packets into an internal 256-byte (per channel) FIFO buffer using a generic 8/16/32-bit wide enhanced UTOPIA level 2 and level 3 compliant interface. HDLC framing performs the insertion of flags, control escape characters and the FCS fields. Either the CRC-CCITT or CRC-32 (in regular or reversed mode) can be computed and added to the frame. Counts of transmitted packets and errored/dropped packets are accumulated for performance-monitoring purposes.

The device is provisioned, controlled, and monitored using a generic 16-bit microprocessor interface. A 4-bit general-purpose input/output (GPIO) interface is provided to control and/or monitor other onboard devices.

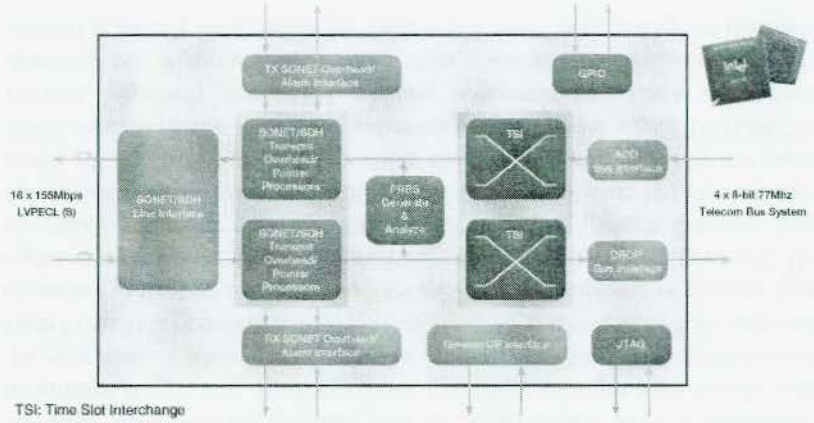


Figure 129: An example – Intel WG1510 (SONET/SDH add/drop multiplexer)

2.2.4 Case Study: Implementation of SPI-4 Interface

The complete interface is bidirectional and consists of four synchronous point-to-point connections. One of these is used for transmitting data (16 bits data, clock 311 MHz, control word signaling), one for the corresponding status channel (2 bits FIFO status, clock 1/4 data clock rate) and two for the other direction. The labels transmit and receive always refer to the payload data flow on the link consisting of data and status channel. All lines belonging to the data channel of the respective direction feature double data rate3 (DDR) transfers. Additionally the electrical parameters of the I/O drivers follow the low-voltage differential signaling (LVDS) standard4. Since the interface is designed as a symmetric one, the same core can be used on both sides of the bus connecting the two protocol layer devices. The regarded part of the network stack for an optical network consists of the application (or link layer) connected via SPI-4 to a physical layer device (e.g. SDH framer). Another interface called SFI (SERDES Framer Interface, defined by the OIF) which connects the SDH framer to the SERDES (serializer/deserializer) chip is used within the physical layer. The connection to the transmission media (fiber) is then established with an electrical-optical transceiver.

To be able to support both bidirectional and unidirectional connections, the transmitting and the receiving interface are designed to be completely independent of each other. This separation is mainly achieved by implementing one out-of-band status channel per direction which realizes the flow control mechanism. Further the data and status channels of each interface have separate clocks (driven by the sender) so they can operate independent of the corresponding data channel.

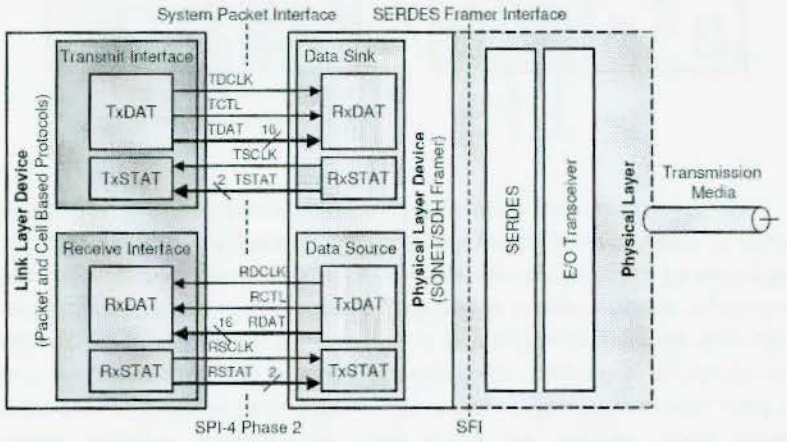


Figure 130: SPI-4.2 interface overview

A common interface for the different transmission standards reduces the effort required for adapting a general network interface card layout to every specification. Ideally this interface should support the communication type's packet and cell transfer and face the problems of board spacing and power consumption. The standardized interface enables the reuse of a general architecture across different line cards and systems.

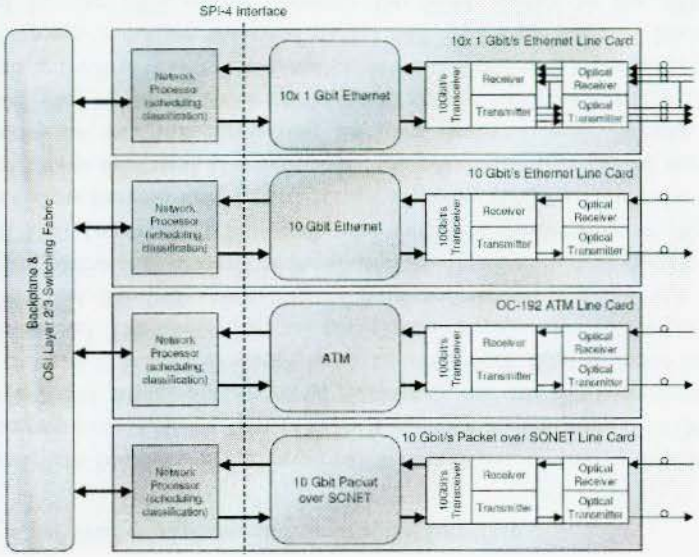


Figure 131: Switch design with SPI-4 interface

In SPI-4, Data is always transferred in bursts which have a determined maximum length and always contain a multiple of eight data words, with the exception of transfers that contain an end-of-packet (EOP) control word. Each burst is associated with a port address, start-of-packet (SOP) and end-of-packet indication and error-control coding. The mapping of packets from higher protocol layers onto the payload stream is shown in Figure 132. The interface supports up to 256 port addresses which allows easy accommodation of 192 SONET/SDH (STS-1 granularity) or 100 Fast Ethernet ports. All additional information about the data is sent in-band by the use of 16-bit control words which are indicated by the special signal TCTL/RCTL. Further, regularly scheduled training sequences are used to perform dynamic bit alignment across the data channel. The in-band signaling method is mainly responsible for the low pin count required by the interface.

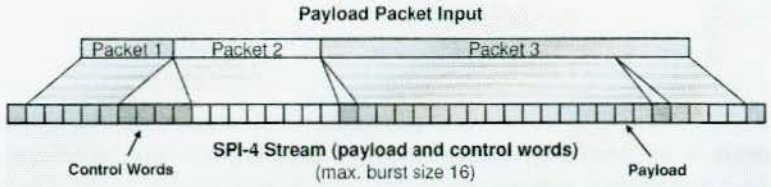


Figure 132: SPI-4 payload mapping

The internal structure of a FPGA is organized into so-called CLBs (configurable logic blocks) which are arranged in an array as shown in Figure 133. The logic function of these blocks can be defined by the user. Further common elements of these chips are internal RAM blocks, multipliers and digital PLLs (phase locked loops). The latter can be used for clock recovery and regeneration (de-skewing), frequency synthesis and phase shifting. Using this functionality it is possible to generate new clock signals derived from another clock signal and/or to regenerate an input clock after transmission. Further these new signals can be varied in phase (which might be used to compensate transmit delays) or period. Often it is also possible for an internal user circuit to change these parameters.

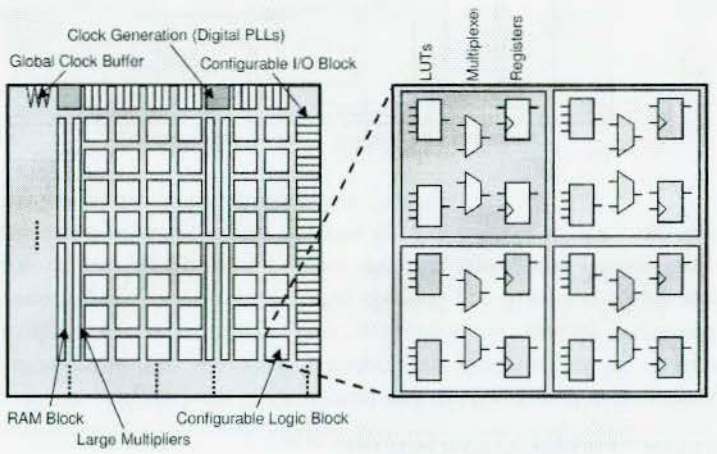


Figure 133: FPGA internal structure (Xilinx Virtex II)

The developed transmit core adds significant new capabilities for design debugging as well as for remote side protocol engine testing by means of

numerous status and error signals, online configurable protocol parameters and various statistical counters. It is also possible to run long-time tests via an external microcontroller, varying several parameters to find the ideal transmission behavior for a given application. This further allows automatic monitoring of the remote side behavior in response to different SPI-4 protocol parameter settings.

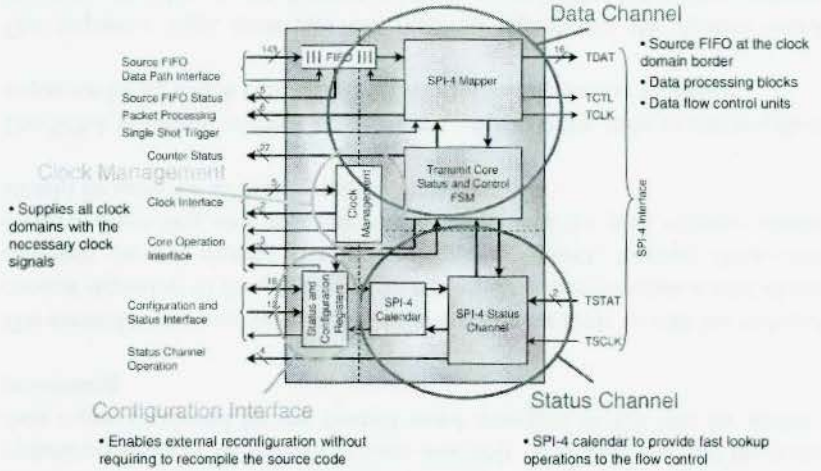


Figure 134: SPI-4 transmit core overview

The transmit core is hierarchically partitioned into single blocks as illustrated in the block diagram shown in Figure 134. The three clock domains separated by appropriate measures depict the highest layer of the design. The data channel consists of the source FIFO at the clock domain border, data processing blocks and the data flow control units. In the current implementation the complete status channel also integrates the SPI-4 calendar in order to provide fast look-up operations to the flow control although the calendar just represents an address area within the configuration registers. While the configuration interface allows external reconfiguration without requiring recompiling the source code, the clock management described at the end of this chapter supplies all clock domains with the necessary clock signals. This unit is responsible for the synchronization of all elements within one clock domain. All other elements

necessary for the information flow over clock borders are implemented on the top source code level.

The data channel uses several pipelined function units to add the SPI-4 in-band control signaling to the payload data stream which is read from the source FIFO. In parallel to the 128-bit bus used to transport the protocol stream an eight-bit bus marks the inserted 16-bit control words which are used to generate the signal TCTL (transmit control) on the SPI-4 interface. Although the core was specified for processing eight words in parallel all units for manipulating the data stream can be configured to use any desired bus width.

The protocol flow is generated in two main steps after reading the payload from the source FIFO and before the data stream is serialized to the interface output. The first module called SPI-4 sequence mapper consists of three single units and inserts the necessary in-band signaling. The second has the same structure but is responsible for the parity checksum securing data transmission.

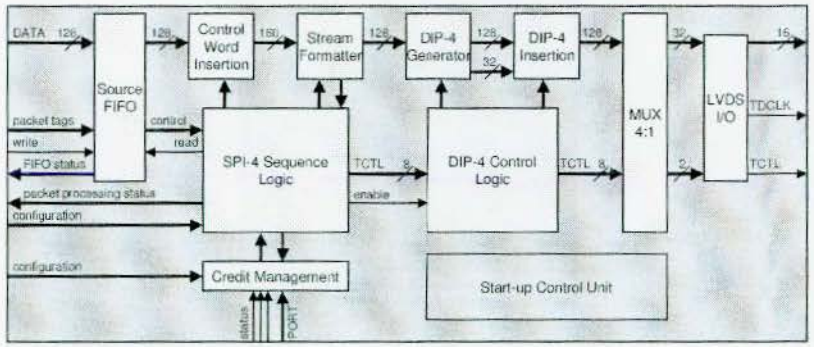


Figure 135: Transmit core data channel structure

The data channel uses several pipelined function units to add the SPI-4 in-band control signaling to the payload data stream which is read from the source FIFO. In parallel to the 128-bit bus used to transport the protocol stream an eight-bit bus marks the inserted 16-bit control words which are used to generate the signal TCTL (transmit control) on the SPI-4 interface. Although the core was specified for processing eight words in parallel all

units for manipulating the data stream can be configured to use any desired bus width.

The protocol flow is generated in two main steps after reading the payload from the source FIFO and before the data stream is serialized to the interface output. The first module called SPI-4 sequence mapper consists of three single units and inserts the necessary in-band signaling. The second has the same structure but is responsible for the parity checksum securing data transmission.

The SPI-4 sequence mapper consists of the three modules sequence logic, control word insertion and stream formatter. The Stream Formatter uses the optimized concept of parallel stream assembly to reduce the 160 bit wide data input produced by the control word insertion to 128 bits for further processing.

The control word insertion is used by the sequence logic to add the in-band control signaling to the payload data stream. The unit requires a total of six different control signals. These are training control, training data, idle control, preceding and adjacent control word and the port number which should be used for all control words in the vector.

The SPI-4 sequence mapper contains one central finite state machine called sequence logic, which controls the SPI-4 protocol operation sequence.

The sequence logic generates the required signals for the control word insertion and the stream formatter which generate the belonging protocol data flow.

The figure shows the state diagram used to generate the protocol flow. After reset the state machine enters the initial training control states. It then toggles between the two states for training patterns. The corresponding training words are generated until the credit management unit indicates valid credit information. Depending on the FIFO status (whether payload is pending for transmission or not) the FSM then switches either to the idle or to the active state. While in state idle the sequence logic just sets idle control words as long as no payload data is waiting, the active state contains the necessary output logic for SPI-4 to conform packet transmission. Payload is sent in bursts up to the currently available amount of credits for the

regarded port. If the optional value MAXBURST was configured to be unequal to zero, the burst length is limited to this number of payload words. Consequently the transfer continues after an interruption using idle control words.

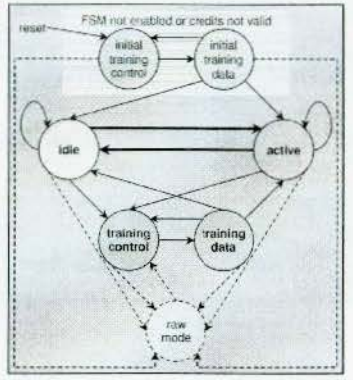


Figure 136: Sequence logic state machine

The chosen structure of the DIP-4 generator represents the serialized algorithm for the parity code. Because there is no need for any registers to store intermediate results, the subunits contain only logic operations. Therefore they do not need to be clocked and it is possible to get a parity output for all stages within one global clock cycle.

The DIP-4 generator block calculates the parity for all eight words in the data stream in parallel.

Because only control words can contain parity, the DIP4 code on the output is only valid if the control word marker is also set. This is also the way how the DIP-4 insertion unit determines the positions where to insert the parity.

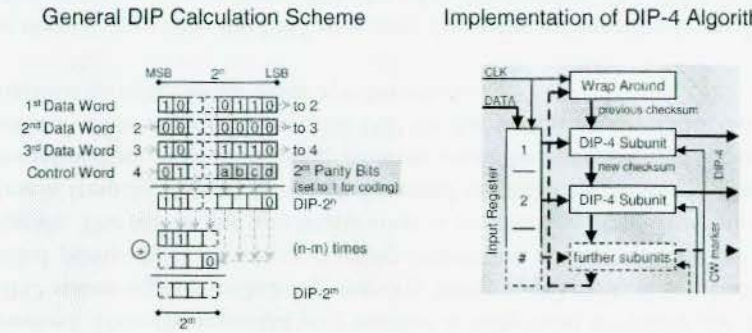


Figure 137: Implemented DIP-4 calculation algorithm

Each subunit takes the checksum of the previous block and adds the corresponding data word of the input register. The DIP-4 parity code for the current stage is used to calculate the new checksum which is then available at the output. In case the input register contains a control word the output checksum is set to all zeros. This is equivalent to a restart of the parity calculation.

Note that only the input and the wrap around register are clocked. Therefore the output should only be sampled at rising edges of the global clock; otherwise the parity data might not be correct due to different latencies in the subunit chain.

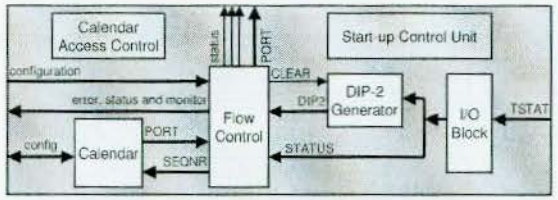


Figure 138: Transmit core status channel structure

The status channel is used to transfer FIFO fill level information from the data sink to the transmitter. It allows the necessary functionality to filter the status information for the configured port numbers out of a multi-port configured status channel transmission.

The start-up control unit is responsible for checking internal conditions (e.g. all calendar entries have been cleared in initialization mode) before setting the appropriate acknowledge signal to the current start-up state.

The calendar is a special memory area containing the sequence of port numbers to be received over the status channel.

The Flow Control entity decodes the remote FIFO fill level information from the status channel, checks the parity information and reports the fill levels to the data path credit management unit.

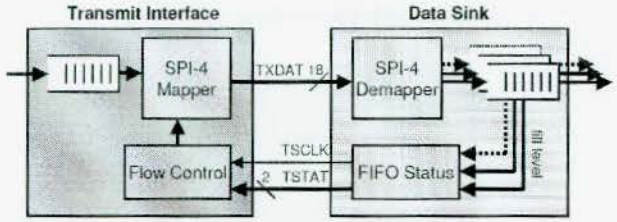


Figure 139: Flow control feedback loop

The feedback loop (shown in the figure for a single source FIFO to a per port sink FIFO implementation) formed by the data path, receiver FIFO and status channel implements a flow control mechanism. In case the receiver application does not retrieve the data fast enough the sink FIFO might overflow. This is avoided by reducing the data burst size on the transmitter side. In order to reduce the incoming amount of data the fill level is reported via three states. Each of them represents a maximum number of payload words which can be sent before the FIFO gets full. The limits are defined in the register fields MAXBURST1 and MAXBURST2. If the remote FIFO is full the transmit data path just sends idle control words. Note that it is necessary to consider the delay of the flow control feedback loop as well as the response time of the sending side when reporting status information.

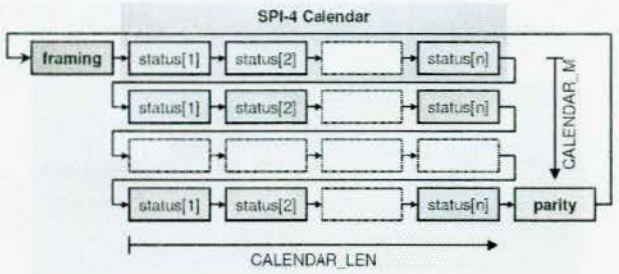


Figure 140: General status channel transmission scheme

Although the core features only one source data buffer it provides multi-port operation and therefore the status channel allows the necessary functionality to filter the status information for the configured port numbers out of a multi-port configured status channel transmission. The overall round-robin period consists of one framing pattern, CALENDAR_M repetitions of the status bits for every port and a final DIP-2 parity. The diagram shows the general case for a transmission on the status channel.

The Flow Control unit is realized using a finite state machine (FSM). After reset the state machine starts in the mode not synchronized and starts searching for correct framing patterns. In this mode the status patterns are not decoded, but the parity and correct position of the framing pattern is checked. Thus the reported port number is only valid if exactly one of the FIFO status signals (indicating starving, hungry, satisfied) is set. Further a failed parity check or wrong framing pattern are not reported by the error signals. The search for synchronization is restarted in state disabled every time a framing pattern is detected instead of status information. After the predefined number of framing patterns could be detected at their correct position (this also requires valid DIP for the period in question) the state machine switches to the mode synchronized.

In normal operation the FSM processes the status information immediately after receiving the pattern. The decoded information including the port number, which is resolved via a look-up to the calendar is reported to the credit management unit of the data channel. In case a framing pattern is detected instead of status information this is considered to be an error and the logic forces a DIP-2 error, independent of the check result at the end of the round-robin period. The state machine reports loss of synchronization

and enters the mode not synchronized either when the predefined number of consecutive DIP errors has occurred or the limit for successive failed framing pattern detections has been reached. Both conditions for leaving the mode synchronized can be disabled by setting the appropriate error limit to zero.

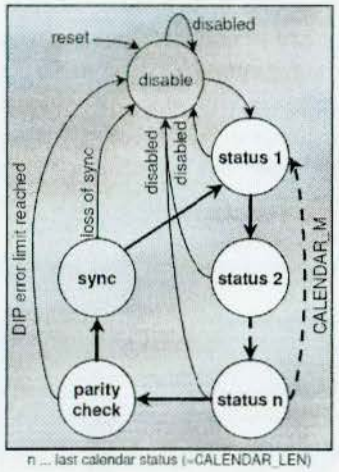


Figure 141: Flow control state machine

Single signals connecting different clock domains are sampled using the double buffering technique. It uses two sequencing flip-flops connected to one single clock source sampling a signal coming from another clock domain as shown in the figure. While it is possible that the first flip-flop is meta-stable, it is assumed that the oscillations fade away within one clock period. Therefore the second sampling element can guarantee a stable output signal. The frequency used for both flip-flop stages should suffice to catch the shortest pulse expected on the monitored signal.
To pass an event signal from one clock domain to another the structure shown in the figure can be used. The two counters represent one pointer each. The difference between them is the number of events that have not been polled. Both counters use the gray code for encoding their state. This ensures that only one bit per clock cycle is changed. Consequently the possibility for meta stability is reduced to only one output signal and the counter's maximum deviation is limited to one in case of an error. In the

worst case the calculated number of events deviates by two for one clock cycle.

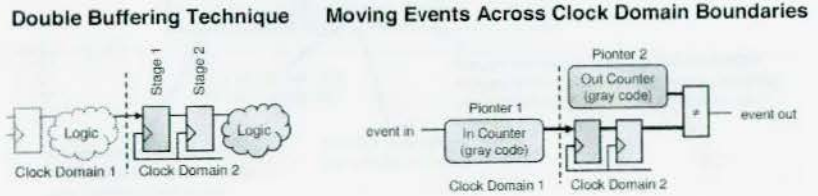


Figure 142: Moving events across domains

Basically a 128:16 serializer could be built by easy programming the multiplexer in behavioral VHDL and connecting this entity to double data rate (DDR) registers. The problem with this approach, using just a behavioral description of the multiplexer, is that the place and route algorithms are not able to optimize the highly parallel design to meet the timing requirements.

The structure shown in Figure 143 contains only two pipeline stages consisting of the input register and the DDR register at the output of the multiplexer. This results in the critical timing path from the input register using the positive clock edge of the slow input clock to the DDR register triggered with the negative edge of the fast transmit clock.

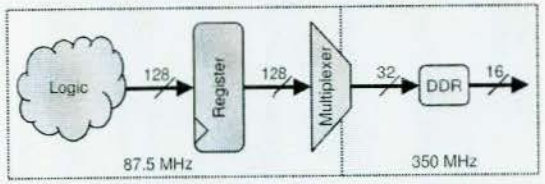


Figure 143: Serializer

By inserting an additional flip-flop at the output of the multiplexer stage the critical timing path is divided into two parts. This approach has two advantages. First the delay caused by the combinatorial logic of the multiplexer is allowed to take up the full transmit clock period. Secondly the allowed delay (one half of the transmit clock period) between the

multiplexer output register and the negative edge triggered DDR register can be fully consumed by the routing delay.

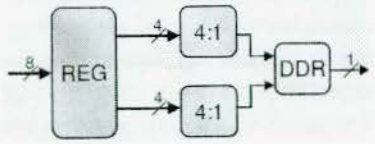


Figure 144: Basic 8:1 multiplexer

By inserting an additional flip-flop at the output of the multiplexer stage the critical timing path is divided into two parts. This approach has two advantages. First the delay caused by the combinatorial logic of the multiplexer is allowed to take up the full transmit clock period. Secondly the allowed delay (one half of the transmit clock period) between the multiplexer output register and the negative edge triggered DDR register can be fully consumed by the routing delay.

The design described above suffices to implement the base element using a transmit clock even higher than 350 MHz. Although the fundamental 8:1 serializer meets the required timing specification, parallel usage of the same unit decreases performance. As shown in Figure 146 the place and route algorithms of common FPGA compilers are not able to keep the timing specification in case multiple base elements are assembled to form a serializer with larger input width.

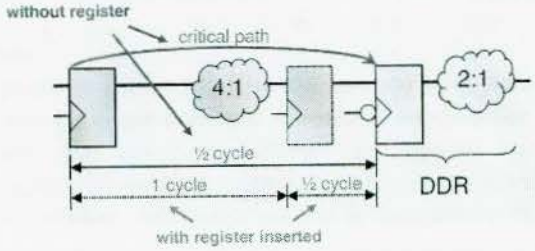


Figure 145: Critical path timing

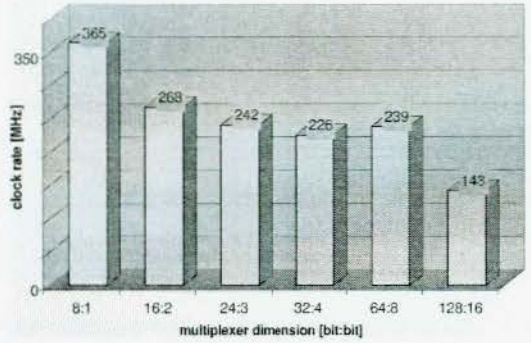


Figure 146: Serializer scalability

To reach an optimal placement it is important to know about the internal structure of the chosen FPGA. The 4:1 multiplexer was designed to fit into one CLB to ensure fast connections between single components. It uses two 4-input LUTs each representing a 2:1 multiplexer with enable and select input which are connected to the built-in 2:1 multiplexer. Further one flip-flop is used for the output register. The elements mentioned so far occupy one slice leaving one flip-flop unused. The second slice of the CLB is used for the counter which uses one LUT (NXOR function) and the two flip-flops for the counter stages. The remaining two slices available in one CLB contain the four input register bits.

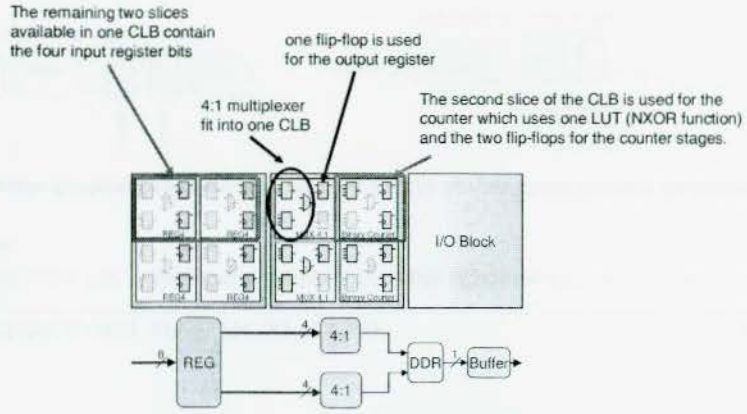


Figure 147: Placement of the serializer

As already mentioned, the whole design is partitioned into sixteen 8:1 multiplexers. This element itself is then divided into the input register, two 4:1 multiplexers and the DDR register. The internal structure of the multiplexers used and the mapping to existing components on the target platform is shown in Figure 148. The combination of all elements is called nibble serializer.

The 4:1 multiplexer itself consists of two look-up tables (LUT) each forming a 2:1 multiplexer with enable input. These two components are then connected to an on-chip multiplexing element. This measure provides lower transport delay due to shorter connection lengths on the chip. In order to keep the routing delay from the binary counter to the multiplexer short, each 4:1 unit has its own binary counter attached. Further using a binary counter facilitates partitioning of the multiplexer into two stages because each counter stage output delivers the select input for one multiplexer stage.

The 128:16 serializer uses thirty-two 4:1 multiplexers connected to the DDR register inputs. Note that the illustrated arrangement of the nibble serializer does not represent the placement of the function blocks on the targeted hardware platform. This meshed interconnection insures that the output bit sequence has the same order as the input bit vector. When used with a 16-bit output, transmission starts with the lowest word of the input vector followed by further six words and the most significant word. After that the next eight words are loaded into the input register of the serializer.

The signal from the nibble serializer connected to the D1 input (triggered with the negative clock edge) of the DDR register is very time critical (one half of the TXLCK period). Thus always two serializers feeding one DDR register are placed together nearby the register to ensure short delays. To reduce the delay on this critical path the multiplexer output registers have been constrained to the immediate vicinity of the I/O block containing the DDR register. Further routing delay reduction is achieved by placing each nibble serializer using four horizontally adjacent slices. This way every serializers of each pair has the shortest connection to the I/O block lying right beside them.

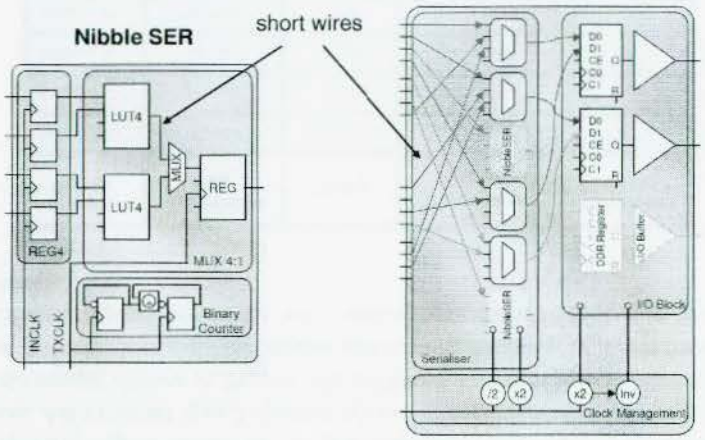


Figure 148: Multiplexer assembly

While the automatically placed design did not scale well with the input data width, the optimized placement provides nearly width-independent performance. Detailed timing simulations show that the new critical path is the connection between the multiplexer output register and the DDR register clocked with the negative clock edge. Because this delay is subject to the chip internal routing paths further speed optimizations can only be achieved using faster target platforms.

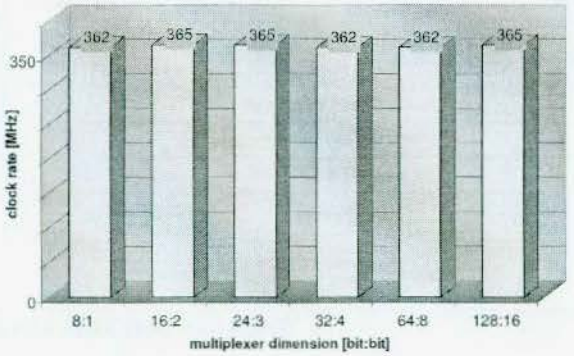


Figure 149: Optimized serializer scalability

2.3 Ethernet Systems

2.3.1 Ethernet Basics

The original Ethernet was developed as an experimental coaxial cable network in the 1970s by Xerox Corporation to operate with a data rate of 3 Mbit/s using a carrier sense multiple access collision detect (CSMA/CD) protocol for LANs with sporadic but occasionally heavy traffic requirements. Success with that project attracted early attention and led to the 1980 joint development of the 10-Mbit/s Ethernet Version 1.0 specification by the three-company consortium: Digital Equipment Corporation, Intel Corporation, and Xerox Corporation (DIX Ethernet).

The original IEEE 802.3 standard was based on, and was very similar to, the Ethernet Version 1.0 specification. The draft standard was approved by the 802.3 working group in 1983 and was subsequently published as an official standard in 1985 (ANSI/IEEE Std. 802.3-1985). Since then, a number of supplements to the standard have been defined to take advantage of improvements in the technologies and to support additional network media and higher data rate capabilities, plus several new optional network access control features.

Figure 150 shows the IEEE 802.3 logical layers and their relationship to the OSI reference model. The OSI data link layer is divided into two sublayers, the Media Access Control (MAC) sublayer and the MAC-client sublayer. The IEEE 802.3 physical layer corresponds to the OSI physical layer.

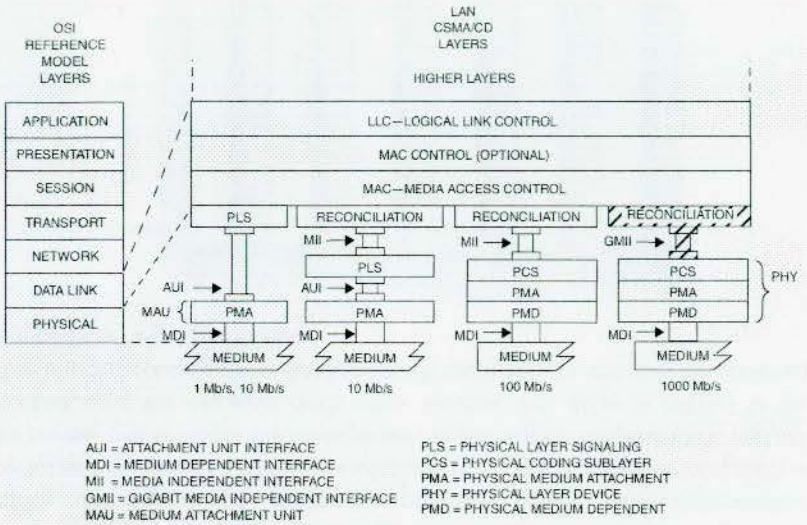


Figure 150: Ethernet (IEEE 802.3)

The specific logical model of the physical layer may vary from version to version. All Ethernet speed options up to Gigabit Ethernet are shown in the generic model shown in Figure 150. Different Ethernet specifications make use of different transmission media and coding schemes. In Table 6 some of the Ethernet specifications for transmission rates from 10 Mbit/s to 10 Gbit/s are shown.

Ethernet Version	Transmission Rate	Coding	Transmission Medium	Full-Duplex Operation ?
10Base-2	10 Mbit/s	Manchester	Coax 50 Ohm	no
10Base-T	10 Mbit/s	4B/5B	Twisted-pair	yes
100Base-T4	100 Mbit/s	8B/6T	Twisted-Pair	no
100Base-FX	100 Mbit/s	4B/5B	Optical Fibers	yes
1000Base-T	1.000 Mbit/s	PAM5	Twisted-Pair	yes
10GBASE-SR	10.000 Mbit/s	64B/66B	Optical Fibers	yes

Table 6: Some examples of Ethernet standards

2.3.2 Ethernet Physical Layer

2.3.2.1 *Transmission Media Options*

Ethernet standards specify several transmission media. The Ethernet signals can be transmitted over copper cables or optical fibers. The most common transmission medium for 100 Mbit/s Ethernet is the unshielded twisted pair (UTP) cable. In this cable, two insulated copper wires are twisted around each other in order to reduce crosstalk or electromagnetic induction between pairs of wires. Other cable options are coaxial cable or optical fiber cable. 10 Gigabit Ethernet standards only specify optical fiber as transmission medium.

For example, 10 Megabit Ethernet can be implemented with these media types:

- 10BASE-2 (Thinwire coaxial cable with a maximum segment length of 185 m)
- 10BASE-5 (Thickwire coaxial cable with a maximum segment length of 500 meters)
- 10BASE-F (optical fiber cable)
- 10BASE-T (ordinary telephone twisted pair wire)
- 10BASE-36 (broadband multi-channel coaxial cable with a maximum segment length of 3,600 meters)

This designation is an Institute of Electrical and Electronics Engineers (IEEE) shorthand identifier. The "10" in the media type designation refers to the transmission speed of 10 Mbit/s. The "BASE" refers to baseband signaling, which means that only Ethernet signals are carried on the medium (or, with 10BASE-36, on a single channel). The "T" represents twisted-pair; the "F" represents fiber optic cable; and the "2", "5", and "36" refer to the coaxial cable segment length (the 185 meter length has been rounded up to "2" for 200).

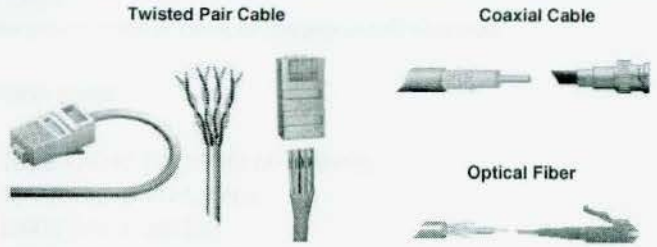


Figure 151: Cabling in Ethernet

Media types specified for different Ethernet standards are listed below:

10BASE-T
- Unshielded Twisted Pair (UTP) cable
- Voice grade cabling
- RJ-45 jack used
- 100 m maximum
- 300 mV signal squelch level (to eliminate cross talk signals), but after 100 m, due to attenuation, data signals also will go below this
- 100 Ω impedance (in some implementations 120 Ω)

100BASE-T
- Unshielded or shielded (for Token Ring) twisted pair
- 100 m, 100 Ω, RJ-45 jack
- 40 pin MII connector also may be used with an external transceiver (not commonly used)
- Data is scrambled to eliminate electro-magnetic effects

100BASE-FX
- 40 pin MII may be used
- If transceiver is built-in, fiber optic can be directly connected
- Non-Return-to-Zero, Invert-on-Ones (NRZI) encoding is used
- Peek optical transmission power is 200 - 400 μW for 62.5/125 μm fiber
- No data scrambling needed
- Two strands of multi-mode fiber optics are used for Tx and Rx

1000BASE-T
- Gigabit Ethernet twisted pair – defined by 802.3ab
- UTP all 4 pairs are used – requires CAT-5 or higher quality cables
- All patch panels also should be of high quality
- Each pair has Tx and Rx wires (total of four Tx and four Rx wires in a cable)
  - hence total of 8 bit at a time
  - 125 Mbaud achieves 1000 Mbit/s
- A combination of signaling and encoding is used to achieve the speed
- No external transceivers or MIIs available - requires a built in transceiver
- Digital Signal Processing (DSP) is used to handle cross talk
- echo cancellation
  - near end cross talk (NEXT) cancellation
  - far end cross talk (FEXT) cancellation
  - signal equalization for distortion compensation
- Auto negotiation possible (10/100/1000 Mbit/s)

1000BASE-CX
- CX – Short Copper Jumper
- 25 m maximum over Twinax Cable
- Used for linking equipment in computer rooms, rack, etc.

1000BASE-SX
- SX: Short Wavelength
- Most widely used
- Less expensive
- Short distance
- Inside buildings

1000BASE-LX
- LX: Long Wavelength
- 500 m typ.
- long haul version up to 10 km
- extended reach version up to 70-100 km

10 Gigabit Ethernet
- Initially only optical transmission medium, but recently there is an effort a new specification over 15 m copper cable (10GBASE-CX4)
- 850 nm over multi-mode fibers (MMF) up to 65 m

- 1310 nm (4 wavelengths WDM up to 300 m over MMF or up to 10 km over single-mode fibers (SMF)
- 1310 nm over SMF up to 10 km
- 1550 nm over SMF up to 40 km

Fiber Type / 10 GbE System	MMF 62.5		MMF 50			SMF
SR/SW - 850 nm	28 m	35 m	69 m	86 m	300 m	-
LR/LW - 1310 nm	-	-	-	-	-	10 km
ER/EW - 1550 nm	-	-	-	-	-	40 km
LX4 - 1310 nm	300 m	300 m	240 m	300 m	-	10 km

Table 7:Maximum distance in 10 Gigabit Ethernet

2.3.2.2 *Coding Methods*

- Baseband Transmission
- Broadband Transmission
- Coding Methods in Ethernet:
  - 10 Mbit/s: Manchester
  - 100BASE-T4: 8B/6T
  - 100BASE-X: 4B/5B
  - 100BASE-T2: PAM-5X5
  - 1000BASE-X: 8B/10B
  - 1000BASE-T: 4D-PAM-5
  - 10GBASE-R, 10GBASE-W: 64B/66B

Manchester Code

The Manchester code is used in the following systems:

- 10BASE-5
- 10BASE-2
- 10BASE-T
- 10BASE-F

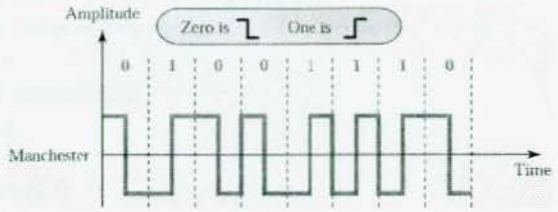


Figure 152: Manchester coding

4B/5B Code

The remaining sixteen 5-bit sequences (not listed in the table) are used for signaling.

4B/5B code is used in the following systems:

- 100BASE-TX
- 100BASE-FX

4-bit Sequence	5-bit Sequence
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Table 8: 4N/5B coding

Non Return to Zero – Invert-on-Ones (NRZ-I) Code

NRZ-I coding scheme is used in the 100BASE-FX.

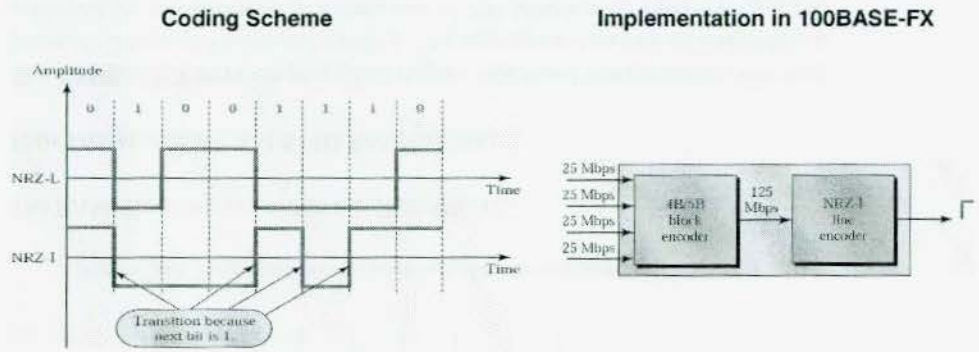


Figure 153: Non-return to zero – inverted on ones (NRZ-I) coding

Multilevel Transmission Encoding - 3 Levels (MLT-3)

MLT-3 uses 3 levels of signals (+1,0,-1). Level transition occurs at the beginning of bit '1'. This code is used to decrease transmission frequency in the 100BASE-TX.

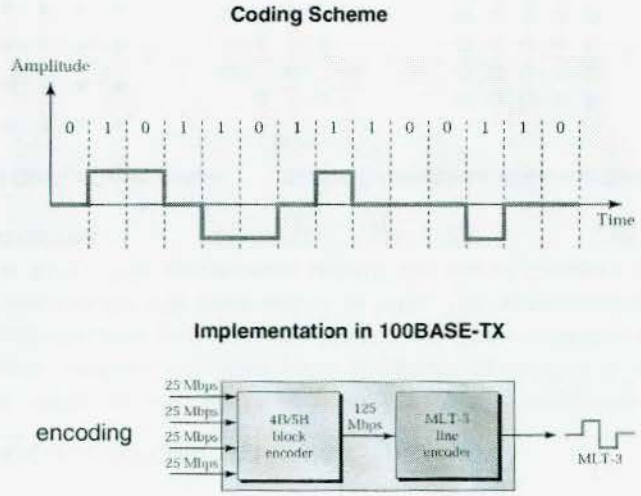


Figure 154: Multilevel transmission encoding – 3 levels (MLT-3)

8B/6T Code

8B6T coding replaces 8-bit data values with six ternary codes, which may have the values '–', '+', or '0'. Table 9 shows a small sample of the 256 code patterns used in 8B6T encoding. The patterns are chosen to provide good DC characteristics, error detection, and reduced high-frequency effects. Special patterns can also be used as markers or control codes. A multilevel signaling scheme is used, which allows more than one bit of data to be encoded into a signal transition. This is why a 12.5 MHz frequency carries a 33.3-Mbit/s stream.

This code is used in the 100BASE-T4.

Bitfolge	8B6T-Code
0000 0000	+ - 0 0 + -
0000 0001	0 + - + - 0
.....	.....
0000 1110	- + 0 – 0 +
.....	.....
1111 1110	- + 0 + 0 0
1111 1111	+ 0 - + 0 0

Table 9: 8B/6T encoding (8 binary/6 ternary)

PAM-5x5 Code

Sending 100 Mbit/s over only two pairs of UTP requires yet another encoding and signaling scheme. In the 100BASE-T2 technology, two five-level pulse amplitude modulation (PAM) signals are sent over the UTP pairs, with a signaling rate of 12.5 MHz. Each cycle of the signal provides two PAM-5x5 level changes, so there are 25 million level changes per UTP pair. Each pair of PAM signals (called A and B) encode a different 4-bit pattern (along with other, special patterns for Idle mode) using combinations of these levels: '+2', '+1', '0', '–1', '–2'. So, 25-million PAM-5x5 pairs × 4 bits per pair = 100 Mbit/s.

Figure 155 show the symbol constellations found in PAM-5x5 encoding, as well as a sample pair of waveforms. Transmitted symbols are selected from the two-dimensional 5 × 5 symbol constellation. Redundancy in the 5 × 5 constellation allows specific encoding rules to be employed to represent data streams received from the interface to the upper layer (medium independent

interface –MII), an idle mode or control signals as sequences of two-dimensional symbols. Five-level pulse amplitude modulation is employed for transmission over each wire pair (PAM 5 × 5). The modulation rate of 25 Mbaud matches the MII clock rate of 25 MHz. The corresponding symbol period is 40 ns. This specification permits the use of Category 3, 4, or 5 balanced cabling.

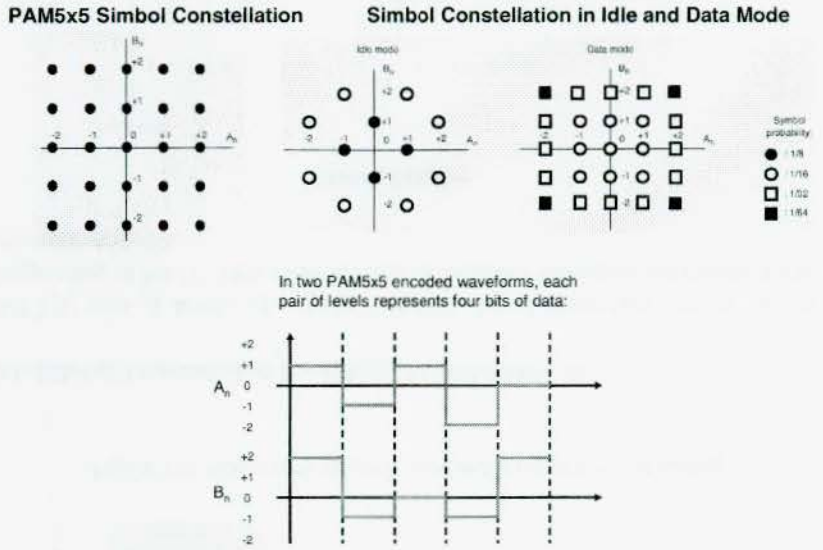


Figure 155: Two dimensional pulse-amplitude modulation (PAM-5x5)

The PAM-5x5 Code is used in the 100BASE-T2.

Four Dimensional PAM-5 (4D PAM-5) Code

The 1000BASE-T PHY employs full duplex baseband transmission over four pairs of Category 5 balanced cabling. The aggregate data rate of 1000 Mb/s is achieved by transmission at a data rate of 250 Mbit/s over each wire pair, as shown in Figure 156. The use of hybrids and cancellers enables full duplex transmission by allowing symbols to be transmitted and received on the same wire pairs at the same time. Baseband signaling with a modulation rate of 125 Mbaud is used on each of the wire pairs. The transmitted symbols are selected from a four-dimensional 5-level symbol constellation.

Each four-dimensional symbol can be viewed as a 4-tuple (An, Bn, Cn, Dn) of one-dimensional quinary symbols taken from the set {'+2', '+1', '0', '–1', '–2'}. 1000BASE-T uses a continuous signaling system; in the absence of data, idle symbols are transmitted. Idle mode is a subset of code-groups in that each symbol is restricted to the set {2, 0, –2}to improve synchronization. Five-level Pulse Amplitude Modulation (PAM5) is employed for transmission over each wire pair. The modulation rate of 125 Mbaud matches the GMII (Gigabit Medium Independent Interface) clock rate of 125 MHz and results in a symbol period of 8 ns.

The process of converting data bits into code-groups is called 4D-PAM5, which refers to the four-dimensional 5-level Pulse Amplitude Modulation coding technique. Through this coding scheme, eight bits are converted into one transmission of four quinary symbols. In 4D PAM-5 there are 54 = 625 possible code sequences. To transmit eight bits, 28 = 256 code sequences are required. The redundancy is used to increase transmission quality through choosing well suited codes. Thus, through special designed codes it can be achieved that more than 90% of the frequency spectra lies below 100 MHz.

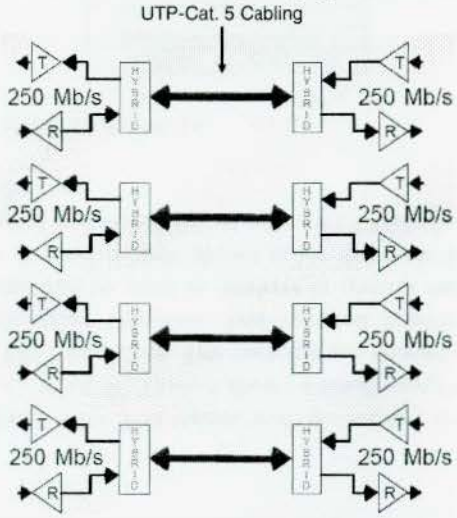


Figure 156: Four dimensional PAM-5 code

Cabling Issues in 1000BASE-T Ethernet

Attenuation is the signal loss of the cabling from the transmitter to the receiver. Attenuation increases with frequency, so designers are challenged to use the lowest possible frequency range that is consistent with the required data rate.

Echo is a by-product of dual-duplex operation, where both transmit and receive signal occupy the same wire pair. The residual transmit signal due to the trans-hybrid loss and the cabling return loss combine to produce an unwanted signal referred to here as echo. Return loss is a measure of the amount of power reflected due to cabling impedance mismatches.

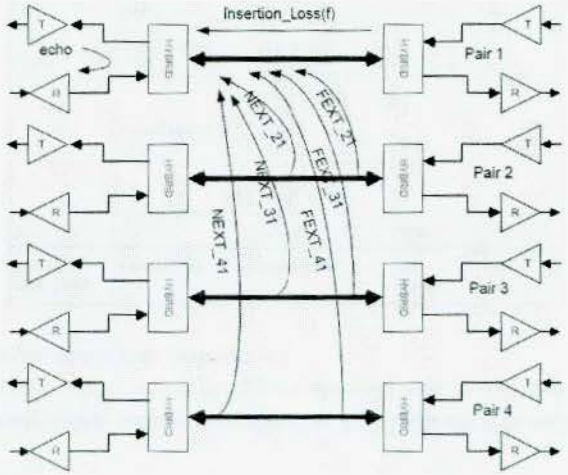


Figure 157: 1000 BASE-T cabling issues

Crosstalk is unwanted signals coupled between adjacent wire pairs. Since 1000BASE-T will use all four wire pairs, each pair is affected by crosstalk from the adjacent three pairs. Crosstalk is characterized in reference to the transmitter. Near-end crosstalk (NEXT) is crosstalk that appears at the output of a wire pair at the transmitter end of the cable and far-end crosstalk (FEXT) is crosstalk that appears at the output of a wire pair at the far end of the cable from the transmitter. Equal level far-end crosstalk (ELFEXT) is FEXT with the cable attenuation removed to provide equal-level

comparisons, i.e., crosstalk and receive signals voltages are compared at the end of the cabling opposite the transmitter. Designers must incorporate technology to keep crosstalk from interfering with symbol recovery operations in the receiver.

8B/10B Code

The 8B/10B transmission code converts a byte wide data stream of random 1s and 0s into a DC balanced stream of 1s and 0s with a maximum run length of 5. The code must also provide sufficient signal transitions to enable reliable clock recovery. A DC balanced data stream proves to be advantageous for fiber optic and electromagnetic wire connections. The average number of 1s and 0s in the serial stream must be maintained at equal or near equal levels. The 8B/10B transmission code constrains the disparity between block boundaries. Certain 10-bit codes in the 8B/10B transmission code have a nonzero disparity value of ±2. These codes require the encoder circuitry to retain the state of the current disparity and select the appropriate ±2 value encoding pair for transmission to maintain DC balance. The coding scheme also implements additional codes for signaling, called command codes.

8B Bytes		10B Codes	
Data Byte Name	Bits HGFEDCBA	CurrentRD- abcdei fghj	CurrentRD+ abcdei fghj
D0.0	000 00000	100111 0100	011000 1011
D1.0	000 00001	011101 0100	100010 1011
D2.0	000 00010	101101 0100	010010 1011
D3.0	000 00011	110001 1011	110001 0100
.	.	.	.
.	.	.	.
.	.	.	.
D28.7	111 11100	001110 1110	001110 0001
D29.7	111 11101	101110 0001	010001 1110
D30.7	111 11110	011110 0001	100001 1110
D31.7	111 11111	101011 0001	010100 1110

Table 10: 8B/10B coding table

The 8B/10B code is used in the following systems:

- 1000Base-LX
- 1000Base-SX
- 1000Base-CX
- 10GBASE-LX4

64B/66B Code

64B/66B transmission code is intended for 10 Gbit/s data transport across a single fiber optic cable. 64-bit blocks are divided into two classes; containing only data, or containing an ordered set.

10B Code Symbol	Description	10B Codes	Begin RD	End RD
/C1/	Link Configuration 1	/K28.5/[1] /D21.5/ config_word1 config_word2	+ or -	flip[2]
/C2/	Link Configuration 2	/K28.5/[1] /D2.2/ config_word1 config_word2	+ or –	same[2]
/C/	Link Configuration	Alternating /C1/ & /C2/	–	–
/I1/	Idle 1	/K28.5/ /D5.6/	+	–
/I2/	Idle 2	/K28.5/ /D16.2/	–	–
/I/	Idle	/I1/ or /I2/	–	–
/S/	Start of packet delimiter (SPD)	/K27.7/	+ or –	same
/T/	End of packet delimiter (EPD)	/K29.7/	+ or –	same
/R/		/K23.7/	+ or –	same
/V/	Error (void)	/K30.7/	+ or –	same

1. config_word 1/2 contain the 16-Bit AutoNegotiation data word. See the AutoNegotiation section for details.
2. RD determined on the /K/ and /D/ characters only, not the config_word.

Table 11: 8B/10B control words

Those blocks containing an ordered set prepend an eight bit code to identify the specific format of the block and map the remaining characters to four, seven or eight-bit codes, depending on block type, to identify the type and order of the ordered sets. In some cases a character is also encoded implicit in the block type identifier (start and terminate characters for instance).

All 64-bits of the all data or ordered set block are then scrambled. A two bit sync header is then added to each 64-bit block; an all data block will receive a 01b sync header and an ordered set block will receive a 10b sync header. The 66-bit block is then ready for transmission.

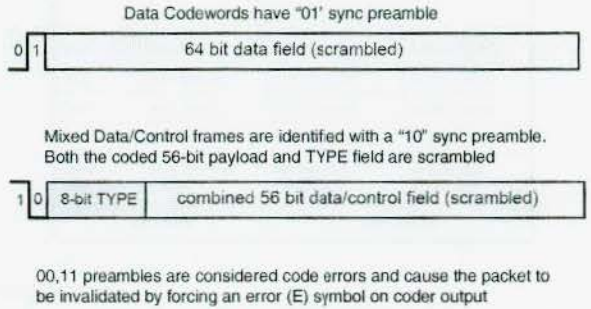


Figure 158: 64B/66B code overview

64B/66B coding scheme encodes 64 bits of data into 66 bits, providing a 3% overhead compared to the 20% overhead of 8B/10B encoding. It improves transmission characteristics of information transferred across link to support control and data characters. Transmission encoding ensures clock recovery is possible at receiver as well as detection of invalid codes when transmitting/receiving. Block alignment is achieved by the synchronization headers.

The 64B/66B Code is used in the following systems:

- 10GBASE-R
- 10GBASE-W

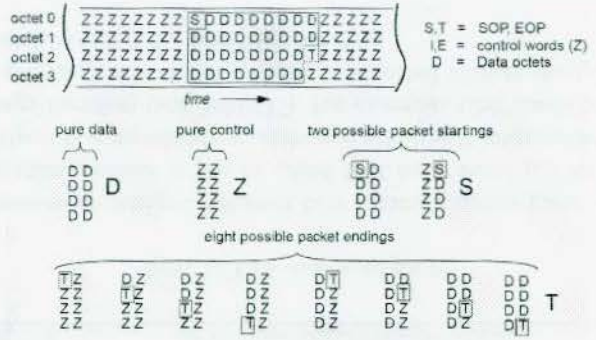


Figure 159: Building frames from 10 GbE reconciliation sublayer (RC) symbols

Input Data	Sync		Bit fields							
$D_0D_1D_2D_3/D_4D_5D_6D_7$	0	1	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$Z_0Z_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x1e	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$Z_0Z_1Z_2Z_3/S_4D_5D_6D_7$	1	0	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
$S_0D_1D_2D_3/D_4D_5D_6D_7$	1	0	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$T_0Z_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0T_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0D_1T_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0xaa	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0D_1D_2T_3/Z_4Z_5Z_6Z_7$	1	0	0xb4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0D_1D_2D_3/T_4Z_5Z_6Z_7$	1	0	0xcc	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0D_1D_2D_3/D_4T_5Z_6Z_7$	1	0	0xd2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0D_1D_2D_3/D_4D_5T_6Z_7$	1	0	0xe1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0D_1D_2D_3/D_4D_5D_6T_7$	1	0	0xff	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Note: all undefined bit fields (in yellow) are set to zero for 10 GbE

Table 12: 64B/66B code summary

8B/10B	name	shorthand	7-bit line code
K28.0	idle1	R	0x00
K28.1	busy idle0	Kb	0x1e
K28.2	reserved0	-	0x2d
K23.7	busy idle1	Rb	0x33
K27.7	start	S	encoded by TYPE byte
K29.7	terminate	T	encoded by TYPE byte
K28.4	reserved1	-	0x4b
K28.5	idle0	K	0x55
K30.7	error	E	0x66
K28.7	reserved2	-	0x78

Table 13: Control code mapping in 64B/66B code

2.3.2.3 *Auto-Negotiation*

The purpose of auto-negotiation is to find a way for two network interface cards (NICs) that share a link to communicate with each other, regardless of the Ethernet version implemented. Fast link pulse (FLP) bursts provide the signaling used to communicate Auto-Negotiation abilities between two devices at each end of a link segment. The inclusion of Auto-Negotiation ensures that the highest performance protocol will be selected based on the advertised ability of the link partner. Following transmission capabilities can be negotiated:

- Communication mode (half duplex or full duplex)
- Transmission speed (10, 100, 10/100)
- Ethernet Version

When the connection is established (plugging both ends of the UTP cable into their respective ports), a series of fast link pulses (FLP) are exchanged between the ports. The 33 pulses contain 17 clock pulses and 16 data pulses. The 16 data pulses form a 16-bit code indicating the capabilities of the port, such as communication mode (half duplex or full duplex) and speed (10, 100, 10/100).

Originally, 10BASE-T NICs used a single normal link pulse (NLP) to perform a link integrity test. NLP pulses are typically generated every 16 ms when the transmitter is idle, as indicated in Figure 160. NICs that support FLPs send a burst containing 2 ms of pulses.

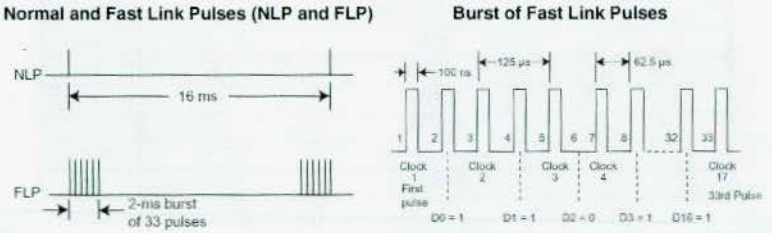


Figure 160: Auto-negotiation process (normal and fast link pulses)

Auto-negotiation protocol is also known as NWay protocol. The operational mode is negotiated in the following priority order:

- 1000BASE-T Full Duplex (Highest Priority)
- 1000BASE-T Half Duplex
- 100BASE-TX full duplex
- 100BASE-TX half duplex
- 10BASE-T full duplex
- 10BASE-T half duplex

A NIC that fails to respond to FLP bursts and returns only NLPs is treated as a 10BASE-T half-duplex NIC.

Bit	Meaning
0 - 4	always 10000 for Ethernet
5	capable of supporting 10BASE-T
6	supports 10BASE-T full duplex
7	capable of supporting 100BASE-TX
8	supports 100BASE-TX full duplex
9	capable of supporting 100BASE-T4
10	supports flow control
11	reserved
12	reserved
13	error indication
14	ACK: acknowledgment of an ANP packet
15	NP: next page (additional information follows)

Table 14: Link code word (LCW)

The information exchanged between two communication parts during the auto-negotiation process is the so called link code word (LCW) shown in Table 14. There is a possibility to exchange additional information by using the next page function (see Table 15). For example, GbE cards can use this feature to exchange additional messages. Asserted 15th in the LCW means that both cards support the next page function.

Bit	Meaning
0 - 10	message block
11	toggle bit
12	ACK-2
13	message bit
14	ACK
15	NP: next page bit

Table 15: Next page function

Meaning of the fields in the next page codewort:

- Message bit: asserted (1): a message page is transmitted; bits 0 to 10 carry a message defined by IEEE committee
- Message bit: de-asserted (0): an unformatted page is transmitted; bits 0 to 10 carry propriety information (manufacturer or card specific)
- Toggle bit: used to distinguish between consecutive pages (different values for two successive pages)
- ACK-2 bit: used to acknowledge a massage block
- ACK bit: used to confirm the reception of the previous code words
- Next-Page bit: shows that additional "next pages" follow

2.3.2.4 *Implementation of the Physical Layer*

A functional block diagram of IEEE 802.3 1000BASE-X specification of the physical layer functions for Gigabit Ethernet is shown in Figure 161. There are three sublayers: physical medium dependant (PMD), physical medium attachment (PMA) and physical coding sublayer (PCS). The functions specified for the PCS are auto-negotiation, synchronisation at the byte level, carrier sense as well as transmit and receive functions. Transmit and receive functions include coder and decoder blocks and interface to the MAC layer by using the Gigabit medium independent interface (GMII).

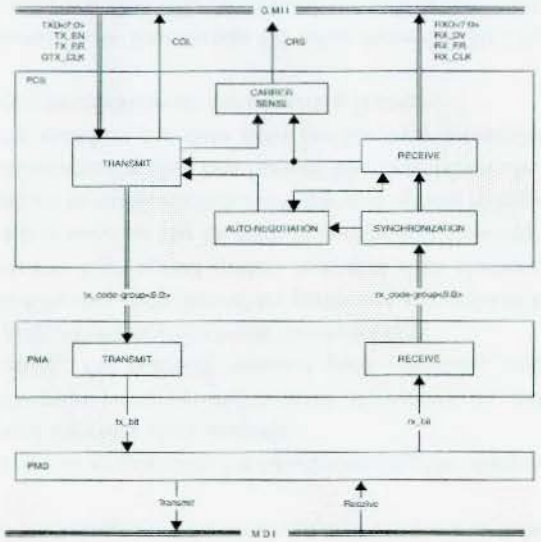


Figure 161: Specification of Gigabit Ethernet physical layer

An example of 10/100 Mbit/s PHY chip is shown in Figure 162. This chip provides all of the IEEE 802.3u 100BASE-TX and ISO 8802-3 10BASE-T physical layer (PHY) functions needed for most workstations, hubs, bridges, router, and switch applications. It can operate as a 100 Mbit/s only device, or as a dual speed 10/100 Mbit/s device with built-in auto-negotiation for speed selection. The chip supports the standard medium Independent Interface (MII) for glueless interface with 10/100 Ethernet medium access controllers (MAC), or MII-based repeaters. It also supports half and full-duplex operation at both 10 Mbit/s and 100 Mbit/s speeds. It includes MLT-3 encoder/decoder and stream cipher scrambler/de-scrambler functions for 100 BASE-TX. The chip supports Category 5 unshielded twisted pair and type 1 shielded twisted pair wiring. The on-chip serial management interface features the basic and extended register set. The 4B/5B encoder and decoder are also included. There are clock generation and clock recovery circuits for both 10BASE-T and 100BASE-TX and requires only one external 25 MHz clock source or crystal. All the clock generation and recovery PLL components are built in to reduce external components and improve noise immunity.

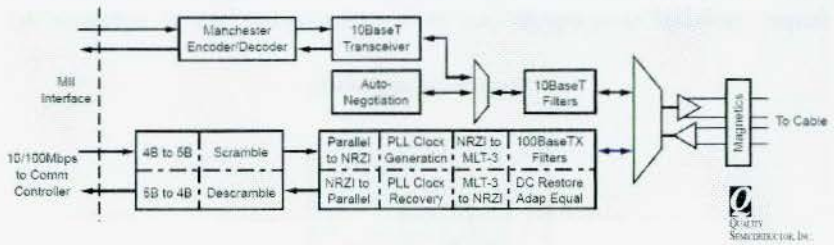


Figure 162: An example of 10/100 Mbit/s PHY chip

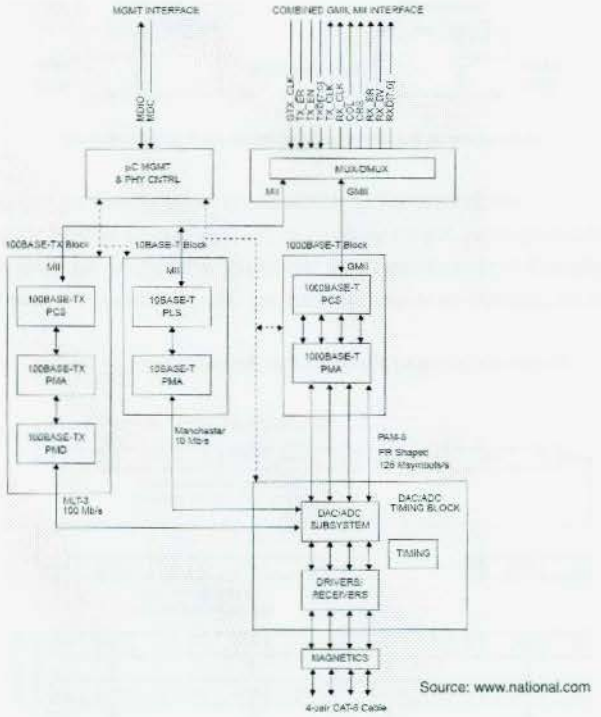


Figure 163: An example of GbE PHY chip

Figure 163 shows a full featured physical layer transceiver with integrated physical media dependent (PMD) sublayers to support 10BASE-T, 100BASE-TX and 1000BASE-T Ethernet protocols. The chip is designed for easy implementation of 10/100/1000 Mb/s Ethernet LANs. It interfaces directly to

Twisted Pair media via an external transformer. This device interfaces directly to the MAC layer through the standard medium independent interface (MII) or the Gigabit medium independent interface (GMII).

A block diagram of the 10 Gigabit Ethernet (10GBASE-R) physical coding sublayer (PCS) is shown in Figure 164. It includes the 10 Gigabit Ethernet scrambler/descrambler and the 64b/66b encoder/decoder.

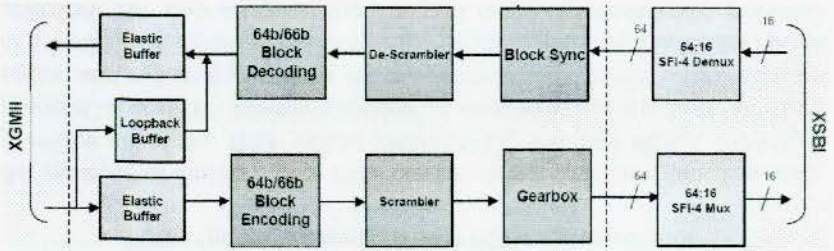


Figure 164: Block diagram of the 10GBASE-R physical coding sublayer (PCS)

- Elastic Buffer is responsible for synchronizing the data packets to the higher speed interface clock domain.
- Encoder/Decoder performs 66-bit word alignment, the 64B/66B receive path decoding, the 64B/66B transmit path encoding, and the 66b/64b transmit path conversion for block overhead bits.
- 10 Gigabit Ethernet data Scrambler generates a transition rich signals to the application high speed optical link and data De-Scrambler on the receive path is used on the TX path to ensure sufficient transitions in the serial interface to support clock recovery and optical transmission.
- Gearbox translates the data bus from 66 bits to 64 bits wide.
- Block Sync searches the data field for the sync preamble in order to perform synchronisation on the incoming frames.

Figure 165 shows how four octets (32 bits) from the 10 Gigabit medium independent interface (XGMII) are scrambled and ordered into eight octets (64 bits) to be encoded in the 64B/66B encoder.

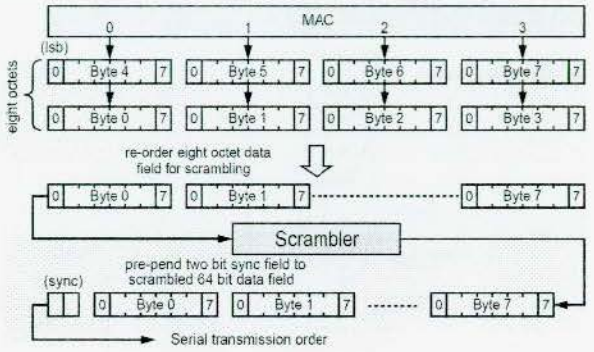


Figure 165: Bit ordering sequence for 64B/66B encoding

Scrambler and de-scrambler can be implemented in parallel or serial form (see Figure 166). IN 10 Gigabit Ethernet, a self synchronizing scrambler with a long pattern length (x58+x39+1=0) is used in order to reduce possibility of jamming. It can be parallelized for efficient implementation.

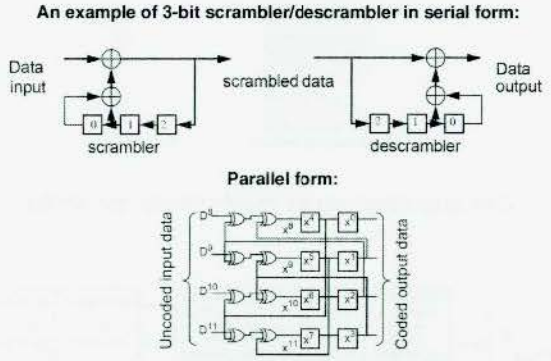
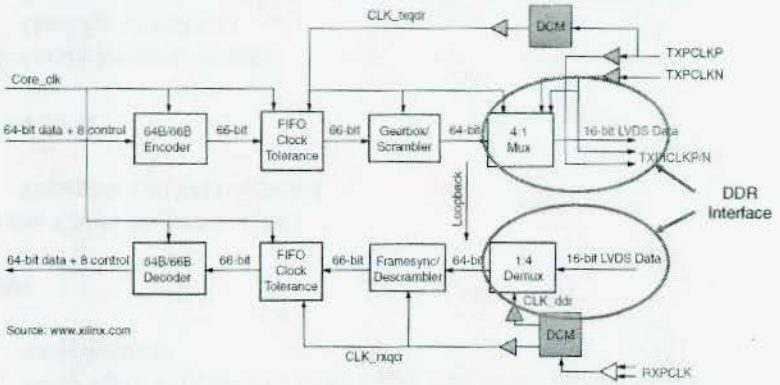


Figure 166: Scrambling

An example of the implementation of the 10GBASE-R physical coding subayer (PCS) in a FPGA device is shown in Figure 167. The implementation consists of Verilog source code, constraint files, and project files to build the design inside application. The interface side already has LVDS (low voltage differential signaling) buffers (IBUFs and OBUFs) for connection to the external 10 Gbit/s transceiver as part of the interface. The MAC side consists

of a 64-bit data bus and 8-bit control bus for each transmit and receive direction. This internal interface can connect directly to the 10 Gigabit Ethernet MAC. Function of the blocks depicted in Figure 167 can be described as follows:

Encoder/Decoder
The encoder and decoder blocks translate from/to the 64-bit XGMII (10 Gigabit Medium Independent Interface) data to 66-bit data bus that is 64B/66B encoded according to the IEEE 802.3ae standard.



	Slices	LUT	Registers	Block RAM	BUFGMUX	DCM
Resource Requirements	4167	6338	3623	8	4	2

DCM: Digital Clock Manager LUT: Look-Up Table DDR: Double Data Rate

Figure 167: An example of 10GBASE-R implementation in a field programmable logical array (FPGA) device

FIFO Clock Tolerance
The FIFO clock tolerance block is responsible for synchronizing the data packets to the higher speed interface clock domain. The write side of the FIFO removes all available idles and (repeated) consecutive sequence ordered sets before writing into the FIFO. The read side inserts idles as needed in order to keep the FIFO half full. The FIFO also holds data for the gearbox and Framesync blocks so that they can properly transmit and receive data. This same block is used on both TX and RX data paths.

Scrambler/Gearbox
The polynomial used for scrambling is specified in the IEEE 802.3ae specification as (1 +x39 +x58). The gearbox is used to translate the data bus from 66 bits to 64 bits wide, so that the 64 bits can be multiplexed into a 16-bit interface.

Descrambler/Framesync
The descrambler is used to reverse the scrambling operation performed on the transmit side, so that the data can be properly decoded. It operates using the same polynomial as the scrambler. The Framesync is used to translate the data bus from a 64-bit non-aligned data bus to a 66-bit word aligned data bus. The alignment is performed by searching the data field for the two Framesync bits.

2.3.3 Medium Access Control (MAC) Layer

The format of the Ethernet 802.3 frame is illustrated in Figure 168. Note that there are also other frame formats used in Ethernet such as 802.3 raw and Ethernet II (or DIX Ethernet) frames. The main difference is that instead of the length field a type field is used that specify the upper layer protocol transmitted in the payload of the frame.

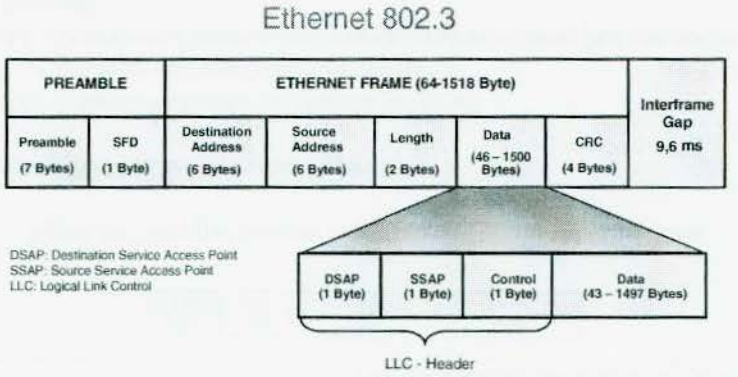


Figure 168: Ethernet frame format

Preamble
- 7 bytes of alternating 0s and 1s – receiver sync
- Not necessary in high-speed Ethernet systems (Fast Eth, GbE)

Start of Frame Delimiter (SFD)
- 10101011 – unique sequence -> last chance to synchronize

Source Address, Destination Address
- 48 bit unique address

Length/Type
- Value up to 1518: length; value larger than 1536 – type of PDU encapsulation

Data

Frame Check Sequence (CRC)
- Preamble and SFD excluded

Data Field

Minimum length is 64 Byte
- Used for CSMA/CD
- Every end station senses the frame within the correct time limits
- Historical requirement derived for bus topology with a coax cable and a network with a 2 byte field

Maximum length is 1518 Bytes
- To assure fair access
- A station should not occupy the medium too long

Frame Bursting in Gigabit Ethernet

Jumbo Frames of 9180 Bytes

A MAC address is a unique value associated with an Ethernet card. MAC addresses are also known as hardware addresses or physical addresses.

I/G	U/L	OUI	OUA
1 Bit	1 Bit	22 Bit	24 Bit

Figure 169: Format of Ethernet medium access control (MAC) address

Format of the MAC address (see Figure 169):

- I/G = 0: Individual address (Unicast Address)
- I/G = 1: Group address (Multicast Address)
- U/L = 0: universal address (worldwide unique address that can not be changed)
- U/L = 1: Local address (can be changed locally)
- Organizationally Unique Identifier (OUI): bits 3 to 24 identify the manufacturer (defined by IEEE)
- Organizationally Unique Address (OUA): bits 25 to 48 are determined by manufacturer

Loss of a frame due to buffer unavailability has the same effect as a frame lost due to a bit error. Effect of frame loss can be devastating on throughput if flow control is implemented by a higher layer protocol. A full duplex Ethernet requires an explicit flow control.
The probability of frame loss due to buffer congestion can be much greater at high data rates. The original Ethernet design did not provide any means for flow control, i.e. ensuring that the senders did not transmit faster than receivers were able to receive. When the network consists of communicating end stations, such mechanisms are normally provided by higher-layer protocols. With the advent of switches/bridges, the immediate receiver may be unknown to the sender.

MAC Control client (such as a switch desiring to prevent buffer overflow) uses the added capabilities of the MAC control sub layer in order to control the data flow and to avoid buffer overflow. It can request the MAC at the other end of a full-duplex link to cease further data transmissions, thereby preventing the impending overflow (PAUSE function).

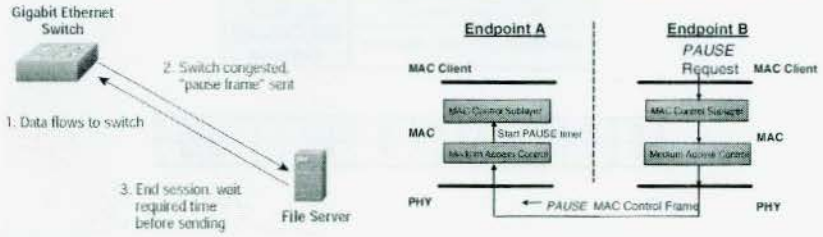


Figure 170: Ethernet flow control (PAUSE frames)

The PAUSE function Implements flow control on a single full-duplex Ethernet link. This is a simple "stop-start" form of flow-control. A device (end station or switch) wishing to temporarily inhibit incoming data sends a PAUSE frame (along with the time to wait). When a device receives a PAUSE frame, it stops sending data frames for the period specified.

The PAUSE function does not solve the problem of steady-state network congestion. The protocol is designed to alleviate temporary overload conditions by reducing inbound traffic in the face of buffer overflow. The PAUSE function does not cure a sustained overload. This mechanism cannot provide end-to-end flow control. It is defined across a single full duplex link only. There is no mechanism provided for either end-to-end flow control or the coordination of PAUSE operations across multiple links.

The specification of the protocol defines what actions occur upon sending or receiving of PAUSE frames, but it says nothing when a device should assert flow control and when it should resume (flow control policy).

The device shown in Figure 171 is a complete media access controller for Gigabit Ethernet with integrated coding logic for fiber and short haul copper media (8B/10B physical coding sublayer). The controller has a 32-bit system interface, receive/transmit FIFO buffers, a full-duplex media access controller (MAC), an 8B/10B PCS, 10-bit physical layer device (PHY) interface and a 16-bit register interface. The controller also contains all the necessary circuitry to implement the IEEE 802.3x flow control algorithm. Flow control massages can be sent automatically without host intervention.

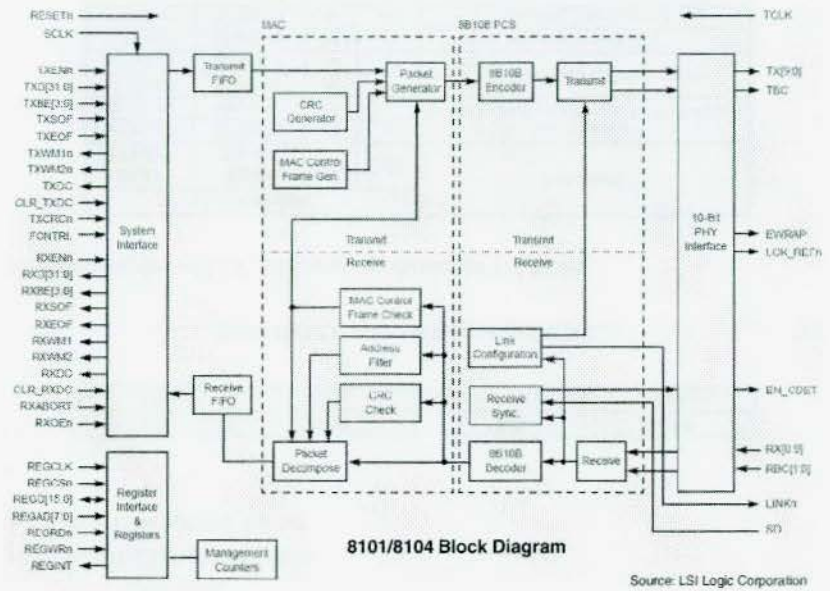


Figure 171: An example of GbE MAC controller implementation

2.3.4 Logical Link Control Sublayer

The IEEE 802.3 standard defines an additional data link layer protocol called the Logical Link Control (LLC) protocol. This operates on top of the MAC protocol defined in the original Ethernet standard. The LLC sublayer presents a uniform interface to the user of the data link service, usually the network layer.

The LLC layer performs following functions:

- Managing the data-link communication
- Link Addressing
- Defining Service Access Points (SAPs)
- Sequencing

There are three types of LLC frames:

- Typ 1 (LLC1): U frames, with an 8-bit control field, are intended for connectionless applications
- Typ 2 (LLC2): I frames, with a 16-bit control and sequence numbering field, are intended to be used in connection-oriented applications
- Typ 3 (LLC3): S frames, with a 16-bit control field, are intended to be used for supervisory functions at the LLC layer.

When the LLC protocol is used, the MAC layer SDU (the payload data) is further encapsulated, adding two additional headers. The extra headers comprise two parts:

- A Logical Link Control (LLC) protocol header
- A Sub Network Access Protocol (SNAP) header

The LLC protocol is based on the HDLC link protocol and uses an extended 2-byte address. The first address byte indicates a Destination Service Access Point (DSAP) and the second address a Source Service Access Point (SSAP). These identify the network protocol entities which use the link layer service (see Figure 168).

A control field is also provided which may support a number of HDLC modes. These include Type 1 (connection-less link protocol), Type 2 (connection-oriented protocol) and Type 3 (connection-less acknowledged protocol).

Bit Nr.	1	2	3	4	5	6	7	8
Typ 1	1	1	M	M	P/F	M	M	M

Bitmuster (Control-Feld)	Funktion des LLC-Rahmens
11000000	UI Command Frame
11111101	XID Command Frame
11110101	XID Response Frame
11001111	TEST Command Frame
11000111	TEST Response Frame

Table 16: LLC control field

Format of the LLC Control Field (Typ 1):

- M: U-Format Coding
- P/F=1: Pool/Final Bit: an acknowledge from the communication part is required

LLC Control Field Typ 1 is used for connectionless services (unnumbered U-Format).

In general, the LLC addresses identify the services of the network layer. Format of the LLC address is as follows:

- I/G = 0: Individual address that identifies a particular DSAP
- I/G = 1: Group address that identifies a group of DSAP addresses or all DSAP addresses
- C/R = 0: Command frame
- C/R = 1: Response frame

I/G	DSAP	C/R	SSAP
1 Bit	7 Bit	1 Bit	7 Bit

Table 17: LLC address format (IEEE 802.2)

Some examples of LLC address are shown in Table 18.

LLC-Adresse				Protokoll
DSAP (Binär)	SSAP (Binär)	HEX	DEZ	
00000000	00000000	00	0	LLC-Instanzen (in Verbindung mit XID- und TEST-Rahmen)
10000000	00000001	01	1	
01000000	00000010	02	2	Management der LLC-Subschicht
11000000	00000011	03	3	
01100000	00000110	06	6	Internet Protocol (IP)
01000010	01000010	42	66	Spanning Tree Protocol
01010101	10101010	AA	170	Subnetwork Access Protocol (SNAP)
00000111	11100000	E0	224	Novell-Protokoll IPX
00001111	11110000	F0	240	Microsoft-Protokoll NetBEUI
01111111	11111110	FE	254	OSI-Protokolle der Netzwerkschicht
11111111	11111111	FF	255	

Table 18: LLC address

Through introduction of SNAP different upper layer protocols (e.g. IP) can be indicated and differentiated. Note that since the maximum size of Ethernet frame is fixed, the maximum size of SDU (payload data) is reduced to 1492 bytes (the MTU in IP) when LLC/SNAP encapsulation is used. This is a similar approach to the protocol number field of DIX Ethernet, which has been replaced by the length field in the 802.3 frame

The SNAP header is transmitted in the payload (data field) of the Ethernet 802.3 frame directly after the LLC header. Format of a SNAP Header is:

- Type: Identification of a network-layer protocol (fully identical with the protocol number field in DIX Ethernet
- Organization Code: Extension of the type fields in order to support new protocol types

Organization Code	Type
3 Bytes	2 Bytes

Table 19: Format of the SNAP header

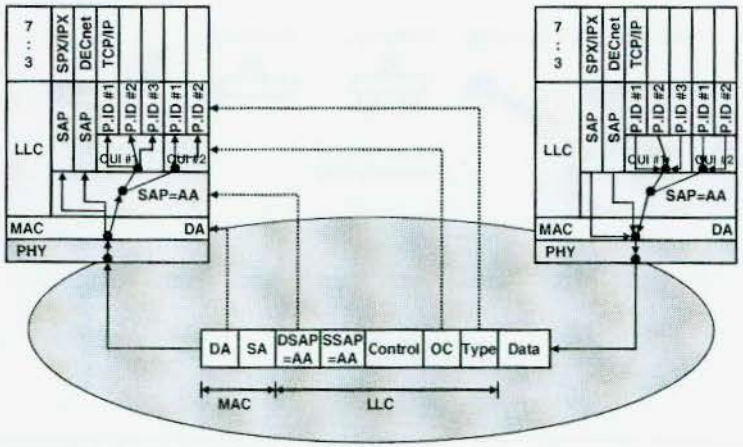


Figure 172: Role of the subnetwork access protocol (SNAP)

2.3.5 Ethernet Interfaces

There are a number of interfaces that are specified and used in Ethernet. They are listed below:

- MAU (Medium Attachment Unit)
- MDI (Medium Dependent Interface)
- AUI (Attachment Unit Interface)
- TBI (Ten-Bit Interface)
- RTBI (Reduced Ten-Bit Interface)
- MII (Medium Independent Interface)
- GMII (Gigabit Medium Independent Interface)
- RGMII (Reduced Gigabit Medium Independent Interface)
- GBIC (Gigabit Interface Converter)
- Hot swappable
- Medium signaling components are contained
- XAUI (10 Gigabit Attachment Unit Interface)
- XSBI (10 Gigabit Sixteen-Bit Interface)
- XGMII (10 Gigabit Medium Independent Interface)

2.3.5.1 Medium Dependent Interface (MDI)

MDI is the physical attachment to the transmission medium. This interface defines physical properties of data signals (e.g. signal level) and mechanical connection (for example construction of a connector). Some of the connector types used in Ethernet are shown in Figure 173.

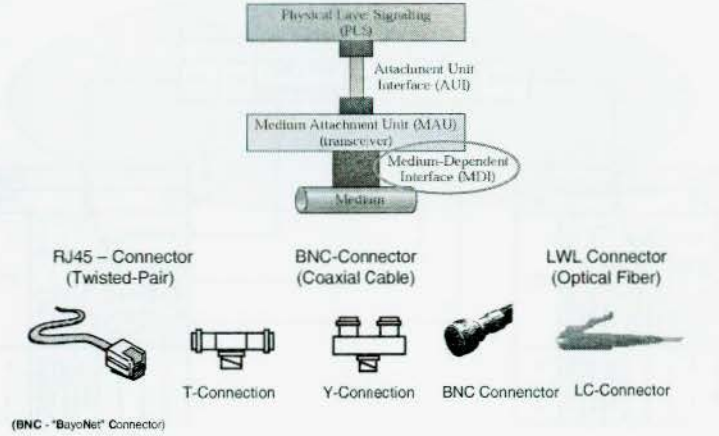


Figure 173: Medium dependent interface (MDI)

2.3.5.2 *Attachment Unit Interface (AUI)*

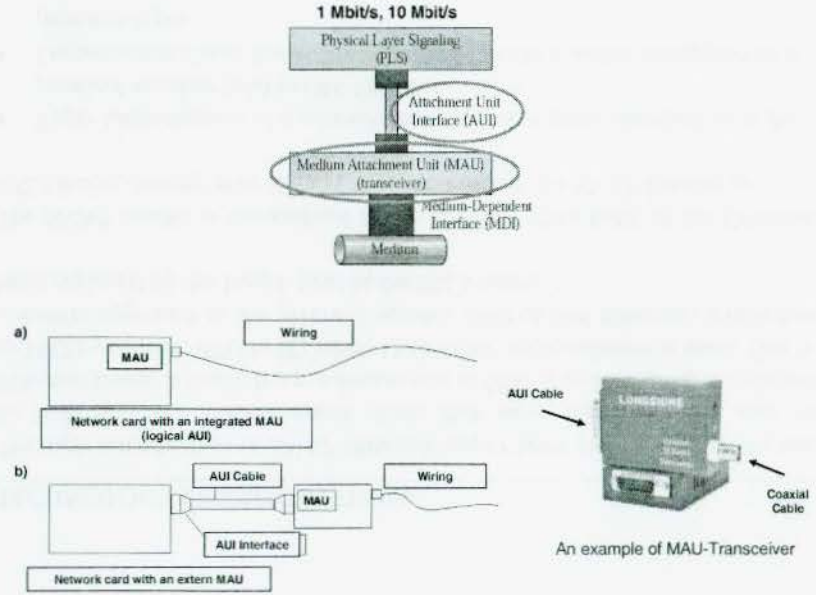


Figure 174: Attachment unit interface (AUI)

The AUI is constructed as a 15-pin connector. 10Base5 envisage a divided medium attachment unit (MAU) in form of a Transceiver (Transmitter-Receiver), which is connected to the AUI over a max. 50 meter long STP cable. Nowadays, the AUI is implemented on the Ethernet network card.

2.3.5.3 *Ten-Bit Interface (TBI)*

TBI is the standard IEEE 802.3 Ethernet Interface, supporting up to 1000 Mbit/s speeds and both full duplex and half duplex operation. It uses two independent clocks, RX_CLK0/1 and TX_CLK, to clock a 10-bit transmit bus (TXD[9:0]) and 10-bit receive bus (RXD[9:0]). Two clock speeds are used: a 125 MHz clock for 1000 Mbit/s Tx interfaces, and two 62.5 MHz clocks for 1000 Mbit/s Rx interfaces. There is another option for ten-bit interface with reduced pin count that supports 5-bit receive and transmit bus that is referred to as reduced ten-bit interface (RTBI). In order to maintain the bandwidth, the data are transmitted on both edges of the clock.

2.3.5.4 *Medium Independent Interface (MII)*

The MII replaces the AUI in Fast Ethernet (100 Mbit/s). The functionality of the MII is the same as those of the AUI, namely to separate MAC layer from different physical layers. It operates at 10 Mbit/s and 100 Mbit/s (backward-compatible).

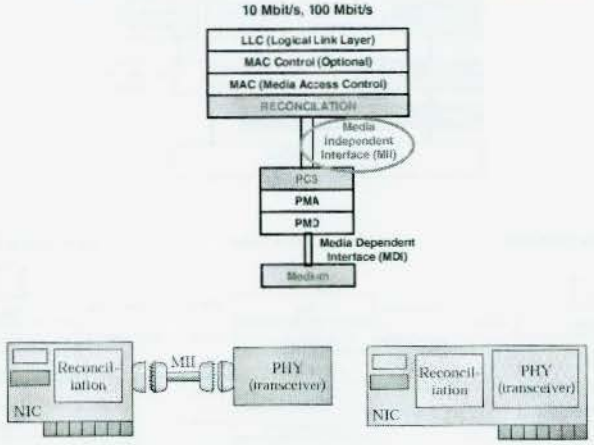


Figure 175: Medium independent interface (MII)

Similar to AUI, MII can be implemented either within a network card or a 40-pin MII connector can be used to externally connect the MAC layer with the physical layer. MII Signals: 4-bit data pins, 25 MHz (or 2.5 MHz for 10 Mbit/s transmission) clock, management signals, collision (COL), carrier sense (CRS), data valid, error und enable. This interface uses two independent clocks, RX_CLK and TX_CLK, to clock a 4-bit transmit bus (TXD[3:0]) and 4-bit receive bus (RXD[3:0]). These two buses are independent from each other as far as the switch is concerned. Two clock speeds are used: a 2.5 MHz clock for 10 Mbit/s interfaces and a 25 MHz clock for 100 Mbit/s interfaces. From the MAC point of view, only transmit data (TXD[3:0]), transmit enable (TX_EN), and transmit error (TX_ER) are outputs, the rest of the signals are inputs. Support of TX_ER is optional for switches. The RX_ER signal needs to be provided by the PHY, but is an optional connect to the switch.

At 100 Mbit/s, Ethernet uses much more complex signal encoding mechanisms than simple Manchester code in 10 Mbit/s Ethernet and these differ from one variant to another. As a result the PMA sublayer is replaced by a more complex, medium-dependent mini-stack known as the PHY. The PHY has three sublayers, the physical coding sublayer (PCS), physical medium attachment sublayer (PMA) and the physical medium dependent sublayer (PMD), which perform the signal encoding and decoding carried

out in 802.3 by the PLS (physical layer signaling) as well as the lower level medium-dependent functions. Because of the variety of options available in 802.3u, including a 10 Mbit/s backward-compatibility mode, encoding and decoding are medium dependent, so the new division of responsibilities is actually quite logical.

A new sublayer added between the MAC layer and MII interface is called reconciliation sublayer. This sublayer replaces PLS sublayer in 10 Mbit/s Ethernet. Encoding and decoding are moved to the PHY (Physical Layer Entity). Reconciliation is responsible for passing data in 4-bit format to the MII (Medium Independent Interface) and to translate the PLS primitives into MIII signals.

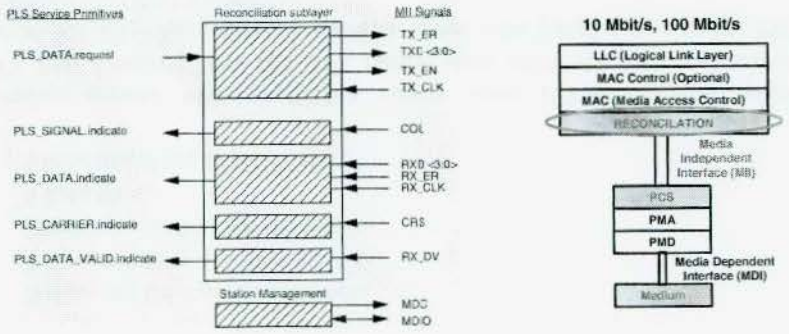


Figure 176: Reconciliation sublayer (RS)

MII management interface is made up of a bidirectional serial data signal, MDIO (management data I/O), and a clock, MDC (management data clock). Optionally, an interrupt pin can be used.

Gigabit Ethernet is a further specification of the IEEE 802.3 standardization group. The reference diagram of Gigabit Ethernet is shown in Figure 177. Note that IEEE has re-used some of the physical layer specifications of the ANSI Fibre Channel. More exactly, they have taken the specifications for FC-0 and FC-1 layers of Fibre Channel and used it in Gigabit Ethernet specifications of 1000BASE-CX and 1000BASE-SX transceivers and 1000BASE-X physical coding sublayer (8B/10B coding).

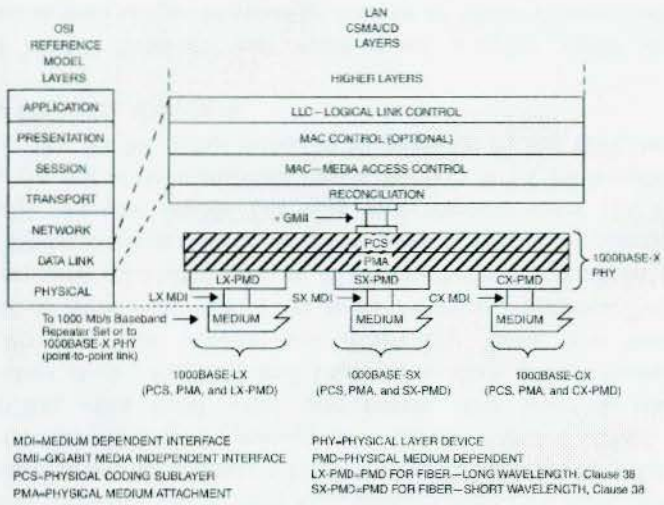
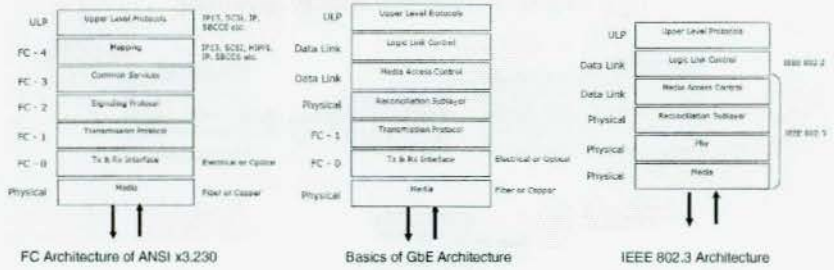


Figure 177: Gigabit Ethernet (IEEE 802.3)



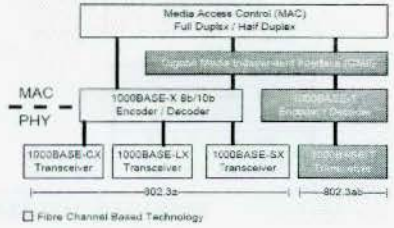


Figure 178: Re-using Fibre Channel specifications in Gigabit Ethernet

Gigabit medium Independent interface (GMII) is the standard IEEE 802.3 Ethernet Interface, supporting 1000 Mbit/s speed and both full duplex and half duplex operation. It uses three independent clocks, RX_CLK, TX_CLK and GTX_CLK, to clock a 8-bit transmit bus (TXD[7:0]) and 8-bit receive bus (RXD[7:0]). These two buses are independent from each other. From the MAC point of view, only transmit data (TXD[7:0]), transmit enable (TX_EN), transmit clock (TX_CLK), and transmit error (TX_ER) are outputs, the rest of the signals are inputs. Support of TX_ER is optional for switches. The RX_ER signal needs to be provided by the PHY, but is an optional connect to the switch. In general, GMII is an extension of MII. Novelties of the GMII are:

- It does not exist outside the NIC
- It operates only at 1 Gbit/s
- There is no connecter or cable
- 8 data pins
- Clock frequency of 125 MHz

Further signals: collision (COL), carrier sense (CRS), transmit enable (TX_EN), transmit error (TX_ER), receive data valid (RX_DV) und receive error (RX_ER). Reconciliation sublayer sends 8-bit parallel data to the PHY sublayer via GMII interface

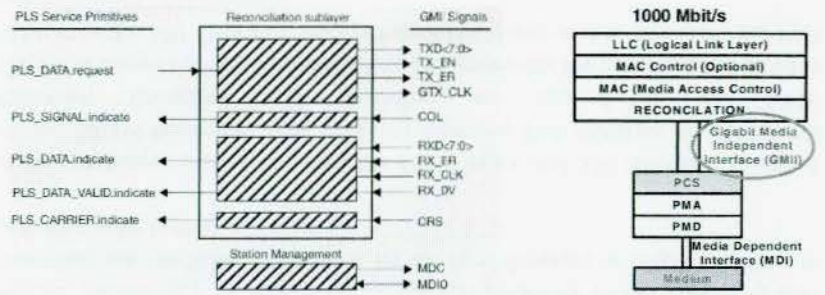
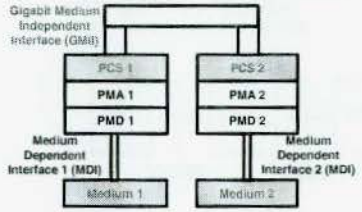


Figure 179: Gigabit medium independent interface (GMII)

In order to reduce the pin count a reduced Gigabit Ethernet media independent interface (RGMII) can be used. The pin reduction is achieved by multiplexing data and control signals on both edges of the reference clocks. Thus, the number of data pins is reduced from 8 to 4.

A GBIC (Gigabit Interface Converter) is a hot-pluggable module based upon the IEEE 802.3z and GBIC specifications v 5.4 for Ethernet and the SFF committee Rev 5.5 specification for Fiber Channel. It is a full duplex serial interface converter that takes electrical signals in a standardized format and converts them into another standardized format to run over fiber or copper cable (ANSI/TIA/EIA-568A).



GBIC	Wavelength (nm)	Fiber Type	Cable Distance
1000BASE-SX	850	MMF	220 m 275 m 500 m 550 m
1000BASE-LX/LH	1300	MMF SMF	550 m 550 m 550 m 10 km
1000BASE-ZX	1550	SMF	70 to 100 km

Figure 180: Gigabit Interface Converter (GBIC)

2.3.5.5 *10 Gigabit Ethernet Interfaces*

The 802.3ae specification defines two PHY types: the LAN PHY and the WAN PHY. The WAN PHY has an extended feature set added onto the functions of a LAN PHY. 10 Gigabit Ethernet uses the IEEE 802.3 MAC sublayer, connected through a 10 Gigabit Medium Independent Interface (XGMII) to Physical Layer entities such as 10GBASE-SR, 10GBASE-LX4, 10GBASE-CX4, 10GBASE-LR, 10GBASE-ER, 10GBASE-SW, 10GBASE-LW, and 10GBASE-EW.

10 Gigabit Ethernet extends the IEEE 802.3 MAC beyond 1000 Mbit/s to 10 Gbit/s. The bit rate is faster and the bit times are shorter - both in proportion to the change in bandwidth. The minimum packet transmission time has been reduced by a factor of ten. A rate control mode is added to the MAC to adapt the average MAC data rate to the SONET/SDH data rate for WAN-compatible applications of this standard. Achievable topologies for 10 Gbit/s operation are comparable to those found in 1000BASE-X full duplex mode

and equivalent to those found in WAN applications. 10 Gigabit Ethernet is defined for full duplex mode of operation only.

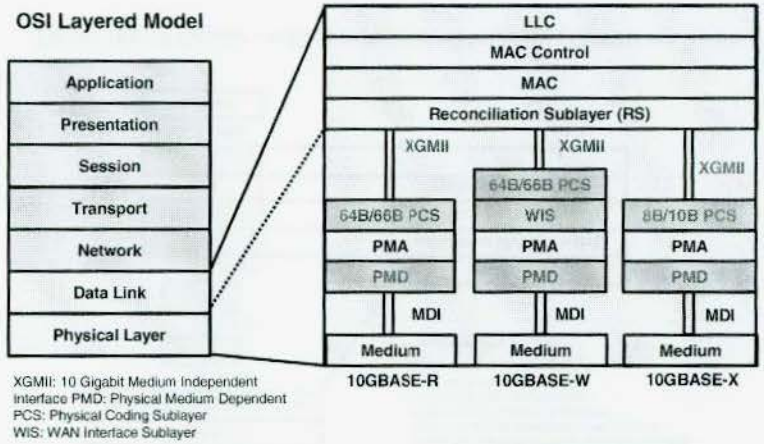


Figure 181: 10 Gigabit Ethernet

The following are the objectives of 10 Gigabit Ethernet:

- Support the full duplex Ethernet MAC.
- Provide 10 Gbit//s data rate at the XGMII.
- Support LAN PMDs operating at 10 Gb/s, and WAN PMDs operating at SONET STS-192c/SDH VC-4-64c rate.
- Allow for a nominal network extent of up to 40 km.
- Supports short wave and long wave optical transmission over multi-mode and single-mode fibers.
- Support operation over 15 m of copper cable.
- Support a BER objective of 10^{-12}.

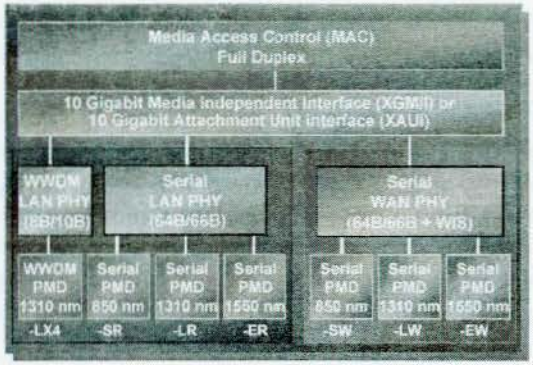


Figure 182: 10 Gigabit Ethernet options

The 10 Gigabit Ethernet interfaces and sublayers are shown in Figure 183.

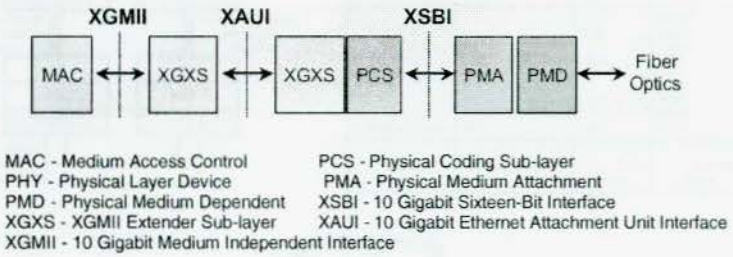


Figure 183: 10 Gigabit Ethernet sublayers

10 Gigabit Medium Independent Interface (XGMII)

Between the MAC and the PHY is the XGMII. XGMII provides full duplex operation at a rate of 10 Gbit/s between the MAC and PHY. Each direction is independent and contains a 32-bit data path, as well as clock and control signals. In total the interface is 74 bits wide (32-bit data paths for each of transmit and receive). While XGMII provides a 10 Gbit/s pipeline, the separate transmission of clock and data results in significant challenge in routing the bus more than the recommended short distance of 7cm. For this reason, chip-to-chip, board-to-board and chip-to-optical module applications are not practical with this interface. Consequently, the XGMII bus puts many

limitations on the number of ports that may be implemented on a system line card.

The XAUI may be used in place of, or to extend, the XGMII in chip-to-chip applications typical of most Ethernet MAC to PHY interconnects.

10 Gigabit Attachment Unit Interface (XAUI)

XAUI (pronounced "zowie") is a full duplex interface that uses four self-clocked serial differential links in each direction to achieve 10 Gbit/s data throughput. The XAUI may be used in place of, or to extend, the XGMII in chip-to-chip applications typical of most Ethernet MAC to PHY interconnects. Each serial link operates at 3.125 Gbit/s to accommodate both data and the overhead associated with coding.

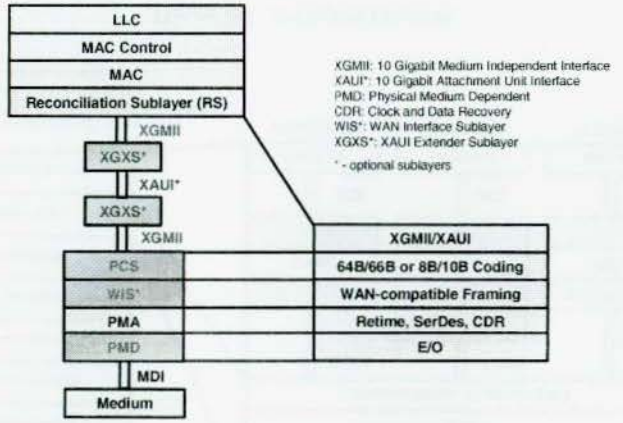


Figure 184: 10 Gigabit medium independent interface (XGMII) extension

The self-clocked nature of the XAUI eliminates skew concerns between clock and data, and extends the functional reach of the XGMII by another 50 cm. Thus, the 74 pin wide XGMII interface is reduced to a XAUI interface consisting of 8 differential pair or 16 pins. In order to convert the XGMII interface signals into XAUI signals an additional sublayer is needed. This function is provided by the so called XAUI extender sublayer (XGXS).

The XAUI employs the same robust 8B/10B transmission code of Gigabit Ethernet to provide a high level of signal integrity through the copper media typical of chip-to-chip printed circuit board traces. Additional benefits of XAUI technology include its inherently low EMI (Electro-Magnetic Interference) due to it's self-clocked nature, compensation for multi-bit bus skew - allowing significantly longer distance chip-to-chip - error detection and fault isolation capabilities, low power consumption, and the ability to integrate the XAUI input/output within commonly available CMOS processes. Mapping of the XGMII signals into the XAUI is shown in Figure 185. The 32-bit XGMII bus is divided into four 8-bit wide groups. Each data byte is then together with one control bit encoded by using 8B/10B coding scheme and transmitted in a serial manner on a XAUI lane.

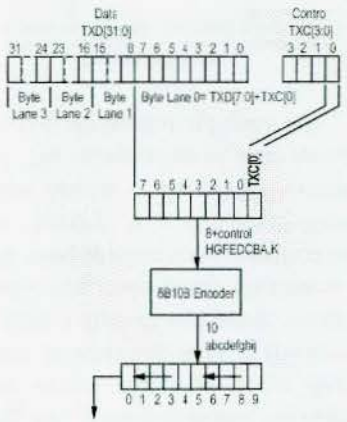


Figure 185: Mapping of XGMII signals into XAUI

10 Gigabit Sixteen-Bit Interface (XSBI)

The 10-gigabit Ethernet XSBI interface is a 16-bit LVDS interface used to connect the physical coding sublayer (PCS) and physical medium attachment (PMA) sublayer. XSBI is based on the Optical Internetworking Forum (OIF) standard SFI-4.

Figure 186 shows variety of interfaces and modules that are needed in different applications of 10 Gigabit Ethernet. Different configurations are

used for wavelength-division multiplexed (WDM) local area networks (LANs), serial LANs, and serial wide area networks (WANs).

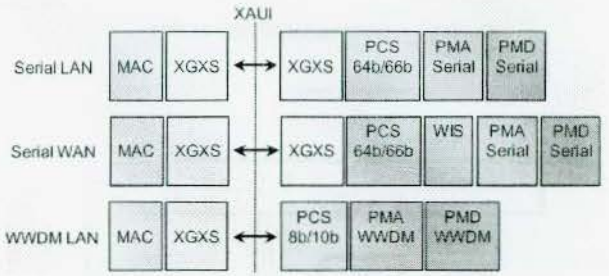


Figure 186: 10 Gigabit Ethernet physical layer (PHY) architecture

WAN interface sublayer (WIS) is a digital wrapper compatible with concatenated SONET/SDH OC-192c/STM-64c systems to provide an Ethernet WAN PHY as an alternative to the LAN PHY for native-format, dark-fiber networks.

A summary of the main characteristics of 10 Gigabit Ethernet interfaces is given in Table 20.

Name	Number of Lines	Data Rate Per Line	Number of Clock and Control Lines	Useful Transmission Distance	Comments
XGMII 10Gb/s Media Independent Interface	32 TX 32 RX	312.5 Mbps	4 Tx Ctrl 1 Tx Clk 4 Rx Ctrl 1 Rx Clk	< 7cm (3")	74 Signal Lines
XSBI 10Gbps Serial Bus Interface	16 TX 16 RX LVDS	622 Mbps	1 Tx Clk 1 Rx Clk	<20 cm (8") + 1 connector	64 Signal lines
XAUI 10Gbps Attachment Unit Interface	4 TX 4 RX Differential	3.125Gbps	None	>50 cm (20") + 2 connectors	16 signal lines

Table 20: 10 Gigabit Ethernet interface summary

2.3.6 Case Study: 8B/10B Encoder/Decoder Implementation

The coding scheme used in IEEE 1000BASE-X and ANSI X3.230-1994 (FC-PH) specifications is a transmission code that improves the transmission characteristics of information to be transferred across the link. The encodings defined by the transmission code ensure that sufficient transitions are present in the PHY bit stream to make clock recovery possible at the receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some of the special code-groups of the transmission code contain a distinct and easily recognizable bit pattern that assists a receiver in achieving code-group alignment on the incoming PHY bit stream. The 8B/10B transmission code specified for use in this standard has a high transition density, is a run-length-limited code, and is dc-balanced. The transition density of the 8B/10B symbols ranges from 3 to 8 transitions per symbol. The relationship of code-group bit positions to PMA and other PCS constructs is illustrated in Figure 160.

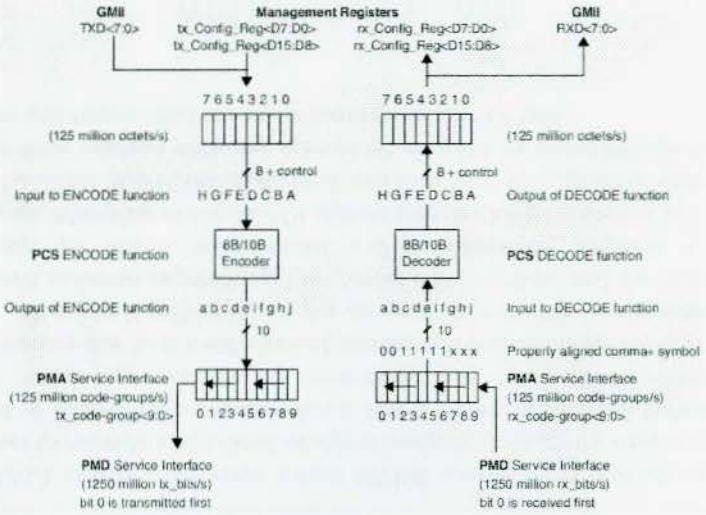


Figure 187: Gigabit Ethernet physical coding sublayer (PCS) reference diagram

8B/10B transmission code uses letter notation for describing the bits of an unencoded information, encoded information, and a single control variable

(Z). The bit notation of A,B,C,D,E,F,G,H for an unencoded information octet is used in the description of the 8B/10B transmission code. The bits A,B,C,D,E,F,G,H are translated to bits a,b,c,d,e,i,f,g,h,j of 10-bit transmission code-groups. 8B/10B code-group bit assignments are illustrated in Figure 188. This figure shows a

Figure 188 shows an 8B/10B encoder block diagram consisting of the eight data lines ABCDEFGH, a control line K, and a clock operating at the byte rate. The control line K indicates whether the lines A to H represent data or control information.

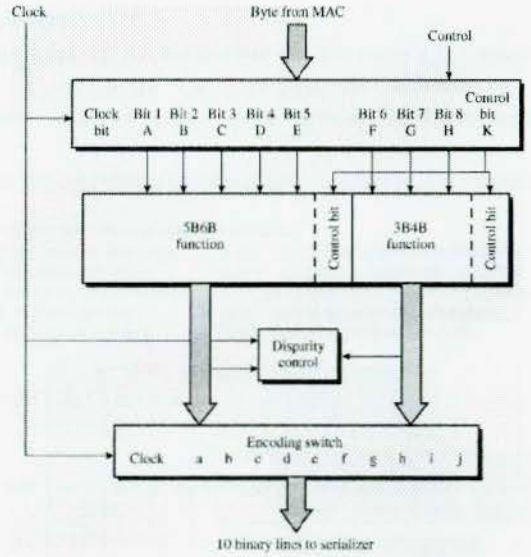


Figure 188: 8B/10B encoder schematic

The 8B/10B encoding scheme is the combination of two sub-block codes, a 5B/6B (ABCDE<=>abcdei) and a 3B/4B (FGH<=>fghj). The 10 encoded bits are serialized by the Gigabit Ethernet transceiver with bit 'a' transmitted first and bit 'j' last. Thus, the encoder is implemented by dividing the 8-bit incoming stream into two sub-blocks of 5-bit and 3-bit length. The five binary lines ABCDE are encoded using the 5B/6B encoding function. Similarly, the three bits FGH are encoded into fghj (3B/4B function).

For both the 6B = abcdei and 4B = fghj sub-blocks the allowed disparity can be 0, +2, or -2. The coding rules require that the polarity of nonzero disparity blocks alternates. For this purpose, a disparity control block has to be added to ensure that the disparity of the encoded sequence is correct.

The design can be segmented into two distinct non-interacting blocks for the transmission (transmitter) and reception (receiver) of 10 bit codes conforming to the IEEE 802.3z specification. These blocks can then be broken down into smaller blocks. The only signal shared between the two blocks is the asynchronous reset input. The overall Encoder/Decoder device block diagram is shown in Figure 189. The de-assertion of the asynchronous reset input to the device is synchronized in two stages for each of the three clock domains. Synchronization allows the full clock cycle in each domain to distribute the asynchronous reset signal to the asynchronous preset or reset of all flip-flops. If the synchronized reset is properly buffered and timing driven place and route is used, the timing can be controlled to make sure the asynchronous recovery requirement is met.

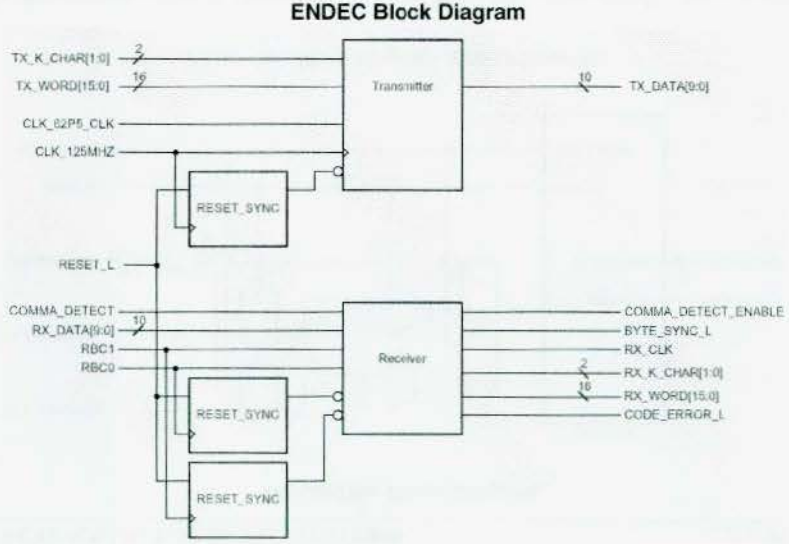


Figure 189: Schematic of an 8B/10B encoder/decoder implementation

The asynchronous recovery requirement is the timing relationship between the asynchronous preset or clear of a flip-flop to the clock input of the flip-flop to avoid metastability. This ensures a clean initialization of the flip-flops. Figure 190 show the logic implementation for the tree-stage synchronization circuit (RESET_SYNC). Note that the circuit has 8 identical outputs for fanout management.

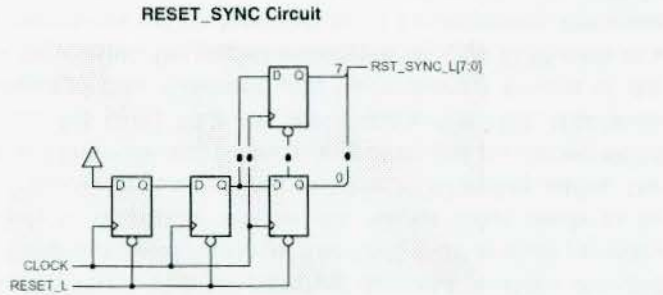


Figure 190: Implementing the RESET_SYNC circuit (synchronous reset)

To reduce the required clock rate for the device providing data for transmission, the transmitter was designed to work off a 16 bit input data bus with two additional inputs to select command or data codes for the high or low byte. This requires an additional 62.5 MHz clock input (TX_62P5_CLK) in phase with the 125 MHz encoder clock. All flip-flops in the transmitter design are clocked with the rising edge of the 125 MHz clock (CLK_125 MHz).

The device sourcing data to the transmitter clocks the 16 bit data and command indicators with the rising edge of the TX_62P5_CLK clock signal, shown in Figure 191. The TX_62P5_CLK input is registered and distributed internally to gain control over the input setup and hold requirements for this signal. The inverter on the TX_62P5_CLK input provides additional delay on the input to reduce the hold time requirement in relation to the CLK_125 MHz input.

Transmitter Block Diagram

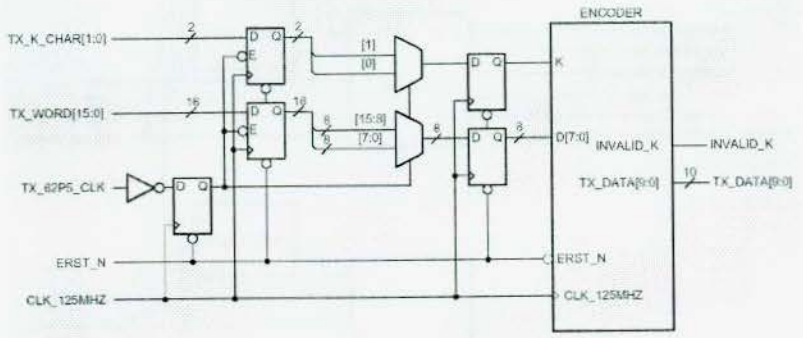


Figure 191: Block diagram of the transmitter

In the encoder block shown in Figure 192 each sub-module does partial encoding of the 8 bit data input value and the command indicator and provides outputs to the third stage of the encoder. The third stage makes the following decisions and provides outputs to the fourth stage:

- Determines whether the data encoding or the command encoding should be transmitted and provides the appropriate 6B and 4B codes.
- Whether or not the 6B code provided requires inversion.
- Whether or not the 4B code provided requires inversion.
- Whether the transmitted encoding should flip the current value of the running disparity or not.
- Pipelined output indicating an invalid command code was requested.

The fourth stage provides the device output registers for the final 8b/10b encoding of the data and a pipelined indicator for an invalid command code request.

Encoder Block Diagram

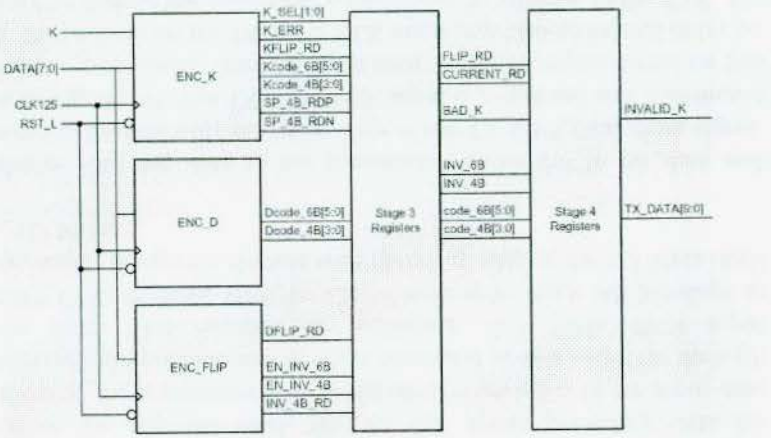


Figure 192: Encoder block

The ENC_D block is responsible for encoding data and provides decoded outputs of the 5B/6B and 3B/4B functions. The ENC_FLIP module provides additional outputs to determine whether the output value needs to be inverted or not. This can be accomplished with two ROM lookup tables. The first ROM table is 4 bits wide and 8 values deep (4x8) and the second ROM table is 6x32. The 4x8 ROM table can use a sequential HDL (Hadware Description Language) case statement with the registered version of the D[7:5] inputs as the selector. The implementation of the 6x32 ROM table in a programmable logic device can be done with six 32-1 multiplexers. For some 3 bit input data values (1, 2, 5, and 6), the 4 bit encoding has only a single value (1001, 0101, 1010, and 0110). In addition, each of the other 3 bit input data values (0, 3, 4, and 7) has 4-bit encoding code pairs. Similarly, for some 5 bit input data values (3, 5, 6, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 25, 26, and 28), the 6 bit encoding has only a single value. The other 4 bit input values have a pair of encoding values dependent on the running disparity. These pairs are the inverse of one another. The ENC_FLIP module provides outputs used by stage 3 to determine if inversions of the 6B value or of the 4B value are needed. Table 21 lists the possible encoding values.

Choosing the 4B values for the 4x8 ROM table

0	=	0100 or 1011	4	=	0010 or 1101
1	=	1001	5	=	1010
2	=	0101	6	=	0110
3	=	0011 or 1100	7	=	0001 or 1110 +

2^3 = 8 different entries

Choosing the 6B values for the 6x32 ROM table

0	=	100111 or 011000	16	=	011011 or 100100
1	=	011101 or 100010	17	=	100011
2	=	101101 or 010010	18	=	010011
3	=	110001	19	=	110010
4	=	110101 or 001010	20	=	001011
5	=	101001	21	=	101010
6	=	011001	22	=	011010
7	=	111000 or 000111	23	=	111010 or 000101
8	=	111001or 000101	24	=	110011 or 001100
9	=	100101	25	=	100110
10	=	010101	26	=	010110
11	=	110100	27	=	110110 or 001001
12	=	001101	28	=	001110
13	=	101100	29	=	101110 or 010001
14	=	011100	30	=	011110 or 100001
15	=	010111 or 101000	31	=	101011 or 010100

2^5 = 32 different entries

Table 21: Encoding of data can be accomplished by using two ROM lookup tables

After first implementation of the encoder total register-to-register delay was 15.7 ns. To work properly the maximum register-to-register delay of the design must be under 8 ns (requirement for 1 Gbit/s TBI). Thus, some data paths have to be divided into smaller units. It can be done by using a pipelined design as shown in Figure 193. Pipelining requires that the implementation device have sufficient usable sequential elements and the architectural freedom of ample routing resources.

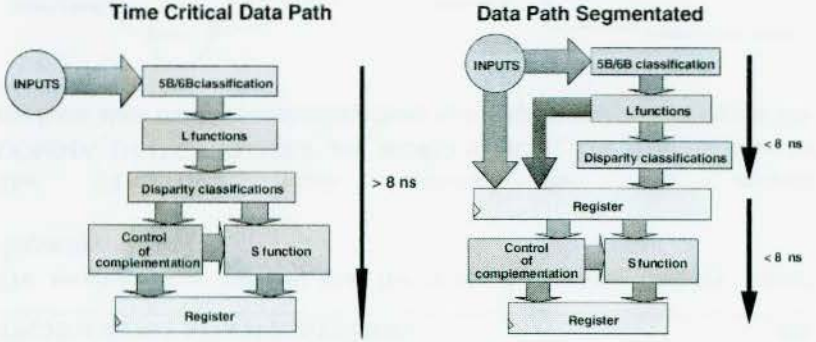


Figure 193: Pipelining of the encoder

The receiver contains two identical decoders, a merge phase block, and a word alignment state machine, as shown in Figure 194. Each decoder circuit works independently off one of the two orthogonal clocks with a clock period of 16 ns. Each decodes the 10 bit input character into its corresponding eight bit value, a K value, eight disparity functions, and provides an indicator for illegal codes. These outputs are then merged into a single clock domain stream of 16 data bits, 2 K values, and a CODE_ERROR_L output. The CODE_ERROR_L can indicate one or both of the following error conditions:

- A character was received with the incorrect disparity. Because detection of a disparity error could potentially be from an earlier received byte and not necessarily from the current byte, the prior three bytes will also be marked as in error with the CODE_ERROR_L output. There are three stages in the merge phase to enable backward indication of code errors when disparity errors occur.
- One of the two bytes in the 16 bit output word was derived from a received code that did not contain a valid 10 bit encoding.

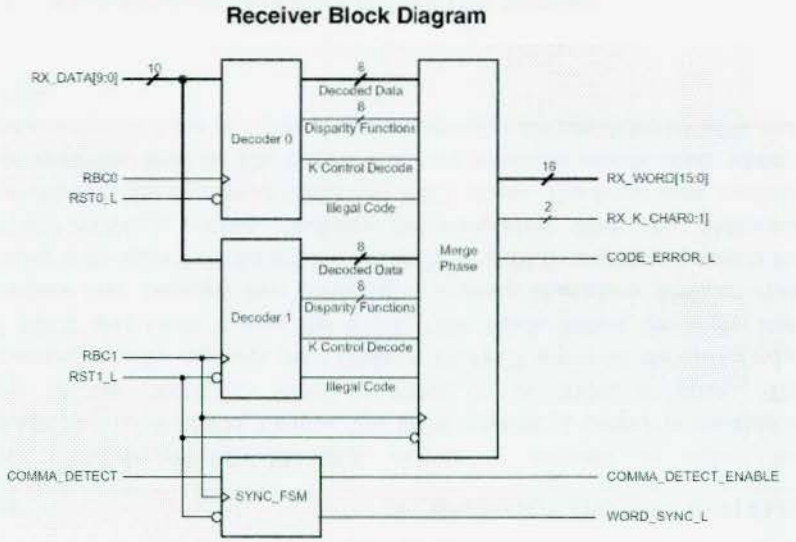


Figure 194: Block diagram of the receiver

The receiver must validate that the received character had the correct disparity.

The SYNC_FSM block controls the two outputs COMMA_DETECT_ENABLE and WORD_SYNC_L. The state diagram for the finite state machine controlling these two outputs is shown in Figure 195.

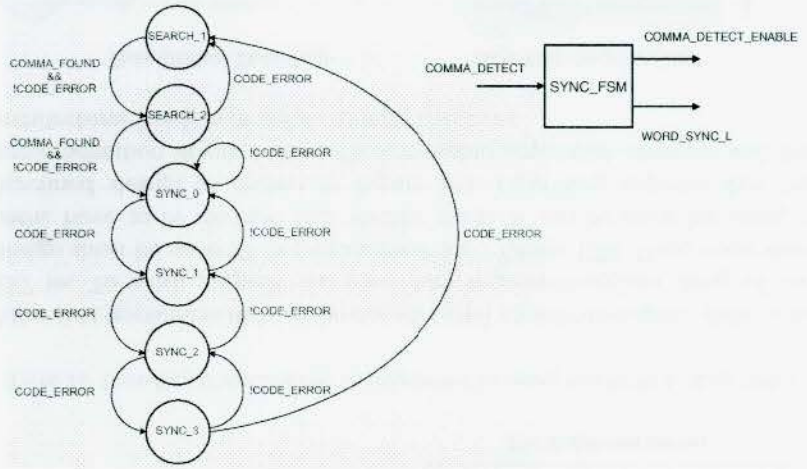


Figure 195: State diagram of the SYNC_FSM block

The Gigabit Ethernet transceiver devices provide 7-bit comma character recognition circuitry that enable the devices to word align the received data stream. The comma character pattern is 0011111XXX, where the leading zero corresponds to the first bit received and the Xs represent don't care bits. Comma characters only occur within the 10 bit command codes K28.1, K28.5, and K28.7. These codes are specifically defined to enable clock synchronization. The word alignment circuitry of the transceiver can be enabled and disabled using the COMMA_DETECT_ENABLE output from the Encoder/Decoder. When the Encoder/Decoder asserts the COMMA_DETECT_ENABLE output, the transceiver scans the incoming data stream for comma characters. When the transceiver finds a comma character it asserts the COMMA_DETECT output and aligns the 10 bit received data with the rising edge of the RBC1 clock.

The COMMA_DETECT_ENABLE output is asserted in either the SEARCH_1 or SEARCH_2 states. The WORD_SYNC_L output is asserted in any of the SYNC_0, SYNC_1, SYNC_2, or SYNC_3 states. The Encoder/Decoder captures the COMMA_DETECT input on the rising edge of RBC1 and feeds it into the SYNC_FSM block. When the finite state machine has detected two consecutive comma characters without code errors, it de-asserts the COMMA_DETECT_ENABLE output and asserts the WORD_SYNC_L output, disabling the transceiver from any additional stretching of the recovered RBC0 and RBC1 clocks. The finite state machine has hysteresis built in. The SYNC_FSM will return to search mode when 4 consecutive, or 5 out of 6 consecutive received codes are detected with code errors.

2.4 Asynchronous Transfer Mode (ATM) Systems

2.4.1 ATM Basics

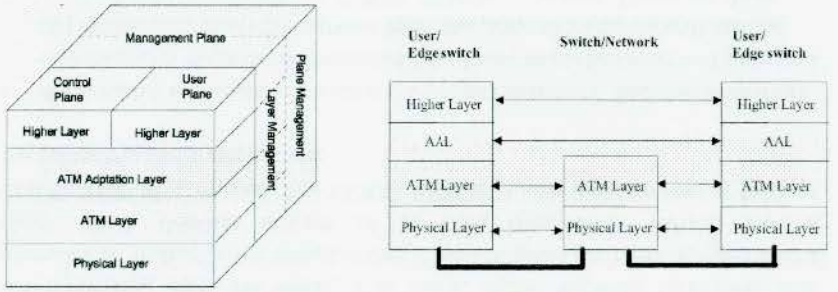


Figure 196: Asynchronous transfer mode (ATM) basics

According to CCITT Recommendation I.150:

- ATM is the transfer mode solution for implementing BISDN.
- ATM uses asynchronous time division multiplexing techniques. The multiplexed information is organized into fixed sized blocks called cells. A cell consists of an information field and a header.
- ATM is a connection-oriented technique.

- In general, signaling and user information are carried on separate ATM layer connections.
- ATM offers a flexible transfer capability common to all services.
- The information field is transported transparently by the ATM layer. No processing, e.g., error control, is performed at the ATM layer.
- The header size and the information field size remain constant at all reference point.

Physical layer provides the physical transport of the ATM cells (cell delineation, header error control, insertion and removal of cells from the physical medium).

ATM layer is common to all services and provides cell transfer functionalities, such as VC/VP (virtual circuit/ virtual path) routing and multiplexing.

ATM adaptation layer (AAL) is service independent and supports higher layer functions of user, control, and management functionalities, such as cell segmentation and reassembly, timing control, flow control. The boundary between the ATM layer and the AAL corresponds to the boundary between functions in the cell header and functions in the cell information field.

ATM protocol separates control and information transfer functions. During VCC setup, only the control plane is active, and during the data transfer, only the user plane is active.

To achieve high speed transport capability, the BISDN network offers minimum functions required to transfer cells. Inside the network, functions performed by the switches for user data transport are limited at ATM layer. AAL functions are provided by the user end devices or the edge switches of the network. In another word, at the user end devices or the edge switches, the higher layer data is adapted into ATM payload by the ATM adaptation layer (AAL) and ATM cells are generated. Inside the network, the switches only examine the ATM cell header and perform ATM layer functions. The information field is transported transparently.

ATM relies on cell-switching technology. ATM cells have a fixed length of 53 bytes which allows for very fast switching. ATM creates pathways between end nodes called virtual circuits which are identified by the VPI /VCI values.

In principle, there are two different cell formats defined. The UNI (user-network interface) or NNI (network-network interface) cell header comprises the first 5 bytes of the ATM cell. The remaining 48 bytes comprise the payload of the cell whose format depends on the AAL type of the cell.

The UNI specification defines communications between ATM endpoints (such as workstations and routers) and switch routers in private ATM networks. The format of the UNI cell header is shown in Figure 197.

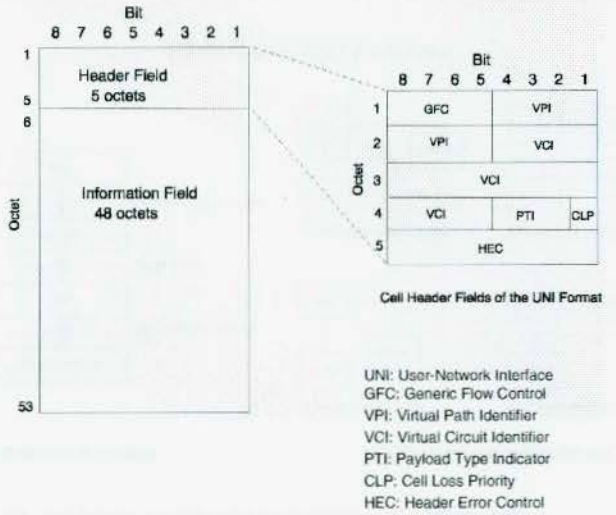


Figure 197: User-network interface (UNI) cell format

The UNI header consists of the following fields:

- GFC: 4 bits of generic flow control that are used to provide local functions, such as identifying multiple stations that share a single ATM interface. The GFC field is typically not used and is set to a default value.
- VPI: 8 bits of virtual path identifier that is used, in conjunction with the VCI, to identify the next destination of a cell as it passes through a series of switch routers on its way to its destination.
- VCI: 16 bits of virtual channel identifier that is used, in conjunction with the VPI, to identify the next destination of a cell as it passes through a series of switch routers on its way to its destination.

- PTI: 3 bits of payload type. The first bit indicates whether the cell contains user data or control data. If the cell contains user data, the second bit indicates congestion, and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.
- CLP: 1 bit of congestion loss priority that indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network.
- HEC: 8 bits of header error control that are a checksum calculated only on the header itself.

The NNI specification defines communications between switch routers. The format of the NNI header is shown in Figure 198.

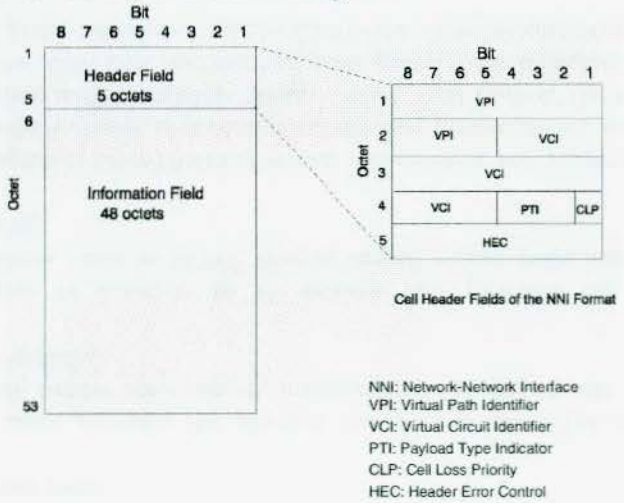


Figure 198: Network-network interface (NNI) cell format

The GFC field is not present in the format of the NNI header. Instead, the VPI field occupies the first 12 bits, which allows switch routers to assign larger VPI values. With that exception, the format of the NNI header is identical to the format of the UNI header.

2.4.2 ATM Interfaces

Universal Test & Operations PHY Interface for ATM (UTOPIA) is defined by the ATM Forum to provide a standard interface between ATM devices and ATM PHY or SAR (segmentation and Re-assembly) devices. The basic reference model for Utopia is represented by Figure 199. This describes functionality for one ATM layer connected to one PHY layer, with an associated management entity. Various Utopia reference configurations based on the reference model are also shown in Figure 199.

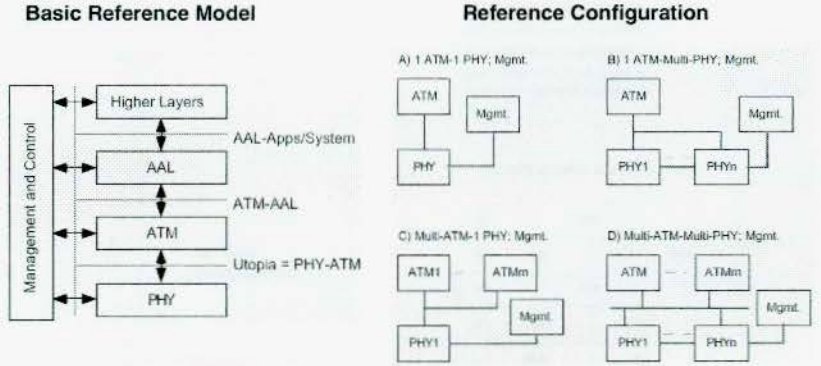


Figure 199: UTOPIA interface

Figure 200 depicts an extended reference model. Both figures show that, for now, the scope of Utopia is limited to the interface between the ATM and PHY layers.

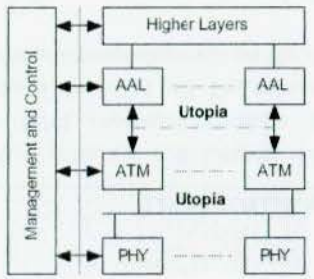


Figure 200: UTOPIA extended reference model

The Utopia standard defines a full duplex bus interface with a Master/Slave paradigm. The Slave interface responds to the requests from the Master. The Master performs PHY arbitration and initiates data transfers to and from the Slave device.

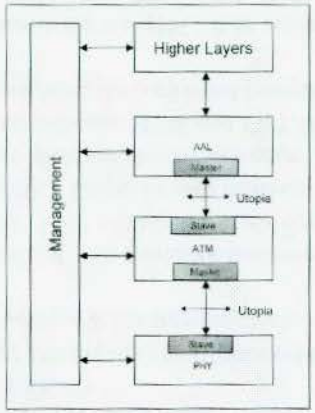


Figure 201: UTOPIA master and slave devices

The ATM forum has standardized the Utopia Levels 1 (L1) to 4 (L4). Each level extends the maximum supported interface speed from OC3, 155 Mbit/s (L1) over OC12, 622 Mbit/s (L2) and OC48, 2.488 Gbit/s (L3) to OC192, 9.952 Gbit/s. The following Table 22 gives an overview of the main differences in these four levels.

Utopia Level 1 implements an 8-bit interface running at up to 25 MHz. Level 2 adds a 16-bit interface and increases the speed to 50 MHz. Level 3 extends the interface further by a 32-bit word-size and speeds up to 104 MHz providing rates up to 3.2 Gbit/s over the interface.

In addition to the differences in throughput, Utopia Level 2 uses a shared bus offering to physically share a single interface bus between one master and up to 31 slave devices (Multi-PHY or MPHY operation). This allows the implementation of aggregation units that multiplex several slave devices to a single Master device. The Level 1 and Level 3 are point-to-point only, whereas Level 1 has no notion of multiple slaves. Level 3 still has the notion of multiple slaves, but they must be implemented in a single physical device

connected to the Utopia Interface. UTPIA Level 4 minimizes the number of signals required on the interface by moving all control functions in-band.

Utopia Level	Interface Width	Max. Interface Speed	Theoretic (typical) Throughput
1	8 bit	25 MHz	200 Mbit/s (typ. STM-1)
2	8, 16 bit	50 MHz	800 Mbit/s (typ. STM-4)
3	8, 16, 32 bit	104 MHz	3.2 Gbit/s (typ. STM-16)
4	8, 16, 32 bit	415 MHz	12.8 Gbit/s (typ. STM-64)

Table 22: UTOPIA level differences

2.4.2.1 *UTOPIA Level 1 Interface*

The definition allows a common PHY interface in ATM subsystems across a wide range of speeds and Medium types. This definition covers connection to devices supporting ATM PHY specifications from sub-100 Mbps to 155 Mbit/s, and provides guidelines for 622 Mbit/s.

The UTOPIA data path specification defines two interface signal groups and general ATM and PHY layer device capabilities. The two interface signal groups are: Transmit and Receive. The device capability specification defines minimum capabilities primarily of the PHY layer device, but secondarily the ATM (SAR) device also.

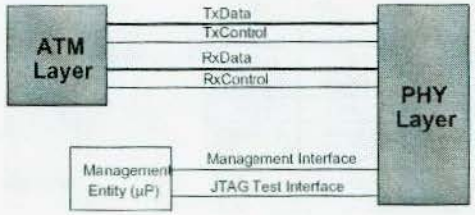


Figure 202: UTOPIA level 1 (L1) specification

Level 1 of the specification covers the following two scenarios:

1. An 8-bit wide data path, using an octet-level handshake, operating up to 25 MHz, with a single PHY device

2. An 8-bit wide data path, using a cell-level handshake, operating up to 25 MHz, with a single PHY device

Utopia level 1 features:

- Transmit and Receive transfers are synchronized via their respective interface transfer clock
- With an 8-bit data path and a maximum clock rate of 25 MHz, this interface supports rates from sub-100 to 155 Mbit/s.

Example interfaces are:

- 155.52 Mbit/s (SDH/STM-1)
- 155.52 Mbit/s (8B/10B block coded)
- 100 Mbit/s (4B/5B TAXI)
- 44.736 Mbit/s (DS-3)
- 51.84 Mbit/s (SONET/OC-1)
- Higher rates (e.g. 622 Mbit/s) may be supported by extending the data path to 16 bits, and using a faster transfer clock.
-

For backward compatibility, cell transfers between ATM and PHY layers provide a "user-defined" field within the data stream. This field may be utilized to transfer the HEC. There are two different cell formats defined for 8-bit and 16-bit interface that are shown in Figure 203. In 16-bit mode, 54-octet cells are transferred between ATM and PHY layers. As for 8-bit mode, a user-defined field is provided for backward compatibility.

If the UDF field is utilized for the HEC, it is recommended that the HEC octet is carried in the UDF1 field. The UDF2 field may be used to provide control over the HEC for test purposes. The field is also provided for backward compatibility to existing devices. HEC error statistics should be measured by PHY layers, and made accessible via a management interface register.

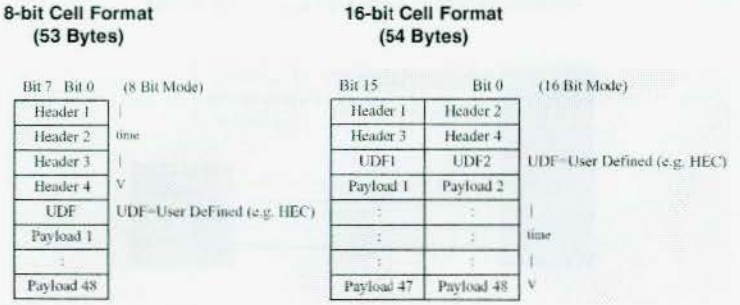


Figure 203: UTOPIA L1 cell format

The following signals are defined as required (R) for the Transmit interface:

- TxData[7..0]: Byte-wide true data driven from ATM to PHY layer
- TxSOC (Start Of Cell): Active high signal asserted by the ATM layer when TxData contains the first valid byte of the cell
- TxEnb* (Enable): Active low signal asserted by the ATM layer during cycles when TxData contains valid cell data.
- TxFull*/TxClav (Full/Cell Available from PHY to ATM): For octet-level flow control, TxFull* is asserted to indicate a maximum of four more transmit data writes will be accepted. For cell-level flow control, TxClav is asserted to indicate that the transfer of a complete cell will be accepted.
- TxClk: Data transfer/synchronization clock provided by the ATM layer to the PHY layer for synchronizing transfers on TxData.

The following signals are defined as optional (O) for the Transmit interface:

- TxPrty (Parity): TxPrty is the odd parity bit over TxData[7:0], driven by the ATM layer.
- TxRef* (Transmit Reference): Input to the PHY layer for synchronization purposes (e.g. 8 kHz marker, frame indicator, etc.).

The following signals are defined as required (R) for the Receive interface:

- RxData[7..0]: Byte-wide data driven from PHY to ATM layer.
- RxSOC (Start Of Cell): Active high signal asserted by the PHY layer when RxData contains the first valid byte of a cell.

- RxEnb* (Enable): Active low signal asserted by the ATM layer to indicate that RxData and RxSOC will be sampled at the end of the next cycle.
- RxEmpty*/RxClav (Empty/Cell Available asserted by PHY): For octet-level flow control, RxEmpty* is asserted to indicate that in the current cycle there is no valid data for delivery to the ATM layer. For cell-level flow control, RxClav is asserted to indicate that there is a complete cell available for transfer to the ATM layer.
- RxClk (Clock): Transfer/synchronization clock from the ATM layer to the PHY layer for synchronizing transfers on RxData.

The following signals are defined as optional (O) for the Receive interface:

- RxPrty (Parity): RxPrty is odd parity for RxData[7:0], driven by the PHY layer.
- RxRef* (Receive Reference): Output from the PHY layer for synchronization purposes (e.g. 8 kHz marker, frame indicator, etc.).

Octet-Level Handshake

During a time period termed the transmit window, the PHY layer stores data from TxData on the low-to-high transition of TxClk, if TxEnb* is asserted. The transmit window exists from the time that the PHY layer indicates it can accept data by de-asserting TxFull*, until 4 valid write cycles after the PHY layer asserts TxFull*. The PHY layer may assert TxFull* at any time, and while asserted this indicates that the ATM layer may transfer no more than 4 data words on TxData. The ATM layer must de-assert TxEnb* within 4 data writes of TxFull* assertion and must not reassert TxEnb* until TxFull* is detected deasserted.4 Asserting TxEnb* outside the transmit window is an error, and the PHY layer will ignore such writes. Inside the transmit window the ATM layer may assert and de-assert TxEnb* as required.

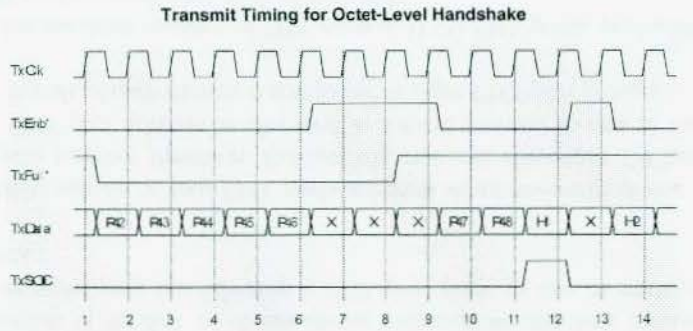


Figure 204: Octet-level handshake

Cell-Level Handshake

The cell-level handshake is identical to the octet-level handshake except for one difference, namely that once TxClav is asserted, the PHY layer must be capable of accepting the transfer of a whole cell. TxEnb* can be used by the ATM Layer to control the flow of data at an octet level (just as for octet-level handshake mode). To ensure that the ATM Layer does not cause transmit overrun, the PHY Layer must de-assert TxClav at least 4 cycles before the end of a cell if it cannot accept the transfer of the subsequent cell.

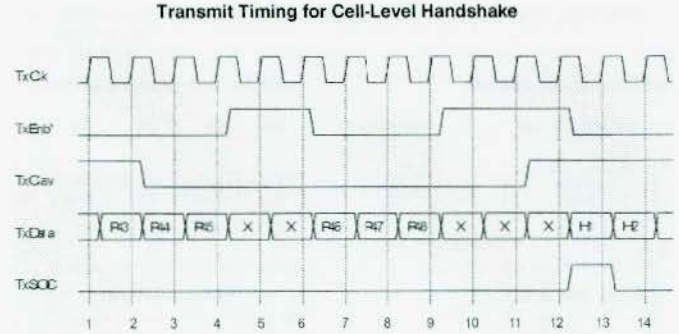


Figure 205: Cell-level handshake

2.4.2.2 UTOPIA Level 2 Interface

In Level 1 Utopia there is only one PHY layer device, and it utilizes TxClav to convey transfer status to the ATM layer. In Level 2 Utopia, only one multi-physical (MPHY) port at a time is selected for a cell transfer. However, another MPHY port may be polled for its TxClav status while the selected MPHY port (device) transfers data. The ATM layer polls the TxClav status of a MPHY port by placing its address on TxAddr. The MPHY port (device) drives TxClav during each cycle following one with its address on the TxAddr lines.

The existing TxFull*/TxClav signal is modified for the Multi-PHY Transmit interface:

- TxFull*/TxClav [0] (Cell Available): For cell-level flow control in an MPHY environment, TxClav is an active high tri-stateable signal from the MPHY to ATM layer. A polled MPHY device (port) drives TxClav only during each cycle following with its address on the TxAddr lines. The polled MPHY device (port) asserts TxClav high to indicate it can accept the transfer of a complete cell, otherwise it deasserts the signal

The additional address bus is defined as required (R) for the Multi-PHY Transmit interface:

- TxAddr[4..0] (Address): Bus driven from the ATM to MPHY layer to poll and select the appropriate MPHY device (port in presence of multiple TxClav signals). Address 31 indicates a null PHY port

The following additional signal is defined as optional (O) for the Multi-PHY Transmit interface:

- TxClav[3..1] (Additional Cell Available Signals): A PHY device (as opposed to a PHY port) may include a total of up to 4 TxClav signals corresponding to 4 PHY ports, which may be used either for direct status indication or for multiplexed status polling

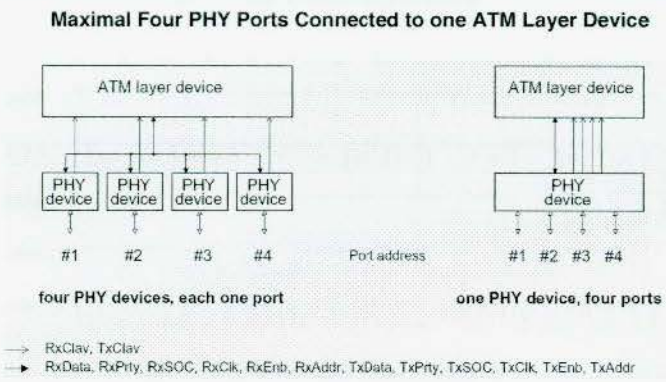


Figure 206: Multiple PHY devices and ports

For each PHY port, the status signals RxClav and TxClav are permanently available according to Utopia Level 1 specification. PHY devices with up to four PHY ports on-chip have up to four RxClav and up to four TxClav status signals, one pair of RxClav and TxClav for each PHY port. In case of less then four PHY ports per device, nevertheless these PHY devices may have up to four status signal pairs. Note, in principle any unique port address can be assigned to the four PHY ports.

Status signals and cell transfers are independent of each other. No address information is needed to obtain status information. Address information must be valid only for selecting a PHY port prior to one or multiple cell transfers.

The status signals of four PHY ports are read simultaneously in one status poll cycle (receive direction: RxClav[3:0], transmit direction: TxClav[3:0]). Every PHY port address is allocated in a fixed manner to one of the four status signals of each direction and to one of eight PHY port groups.

Note, the maximal number of PHY ports is 31 (31 PHY ports, port addresses #0...#30, and one null port, port address #31).

PHY Port Address Allocation

Signal	PHY port Address
RxClav0	0, 4, 8, 12, 16, 20, 24, 28
RxClav1	1, 5, 9, 13, 17, 21, 25, 29
RxClav2	2, 6, 10, 14, 18, 22, 26, 30
RxClav3	3, 7, 11, 15, 19, 23, 27
TxClav0	0, 4, 8, 12, 16, 20, 24, 28
TxClav1	1, 5, 9, 13, 17, 21, 25, 29
TxClav2	2, 6, 10, 14, 18, 22, 26, 30
TxClav3	3, 7, 11, 15, 19, 23, 27

PHY Port Address Allocation to Group Address

PHY Port Address	Group Number	Group Number on Addr[4:0]
0, 1, 2, 3	0	000 xx
4, 5, 6, 7	1	001 xx
8, 9, 10, 11	2	010 xx
12, 13, 14, 15	3	011 xx
16, 17, 18, 19	4	100 xx
20, 21, 22, 23	5	101 xx
24, 25, 26, 27	6	110 xx
28, 29, 30	7	111 0x, 111 10

x = don't care

Table 23: PHY port address allocation

In principle, there is no relationship between PHY port number and PHY port address. To each PHY port one unique PHY port address has been assigned (by configuration, management interface) according to the status signal the PHY device is connected to. For example depicted in Figure 207, PHY device #1 is connected to RxClav0/TxClav0 and its PHY port may have one port address out of [0, 4, 8, 12, 16, 20, 24, 28].

8 PHY Devices, one PHY Port per Device

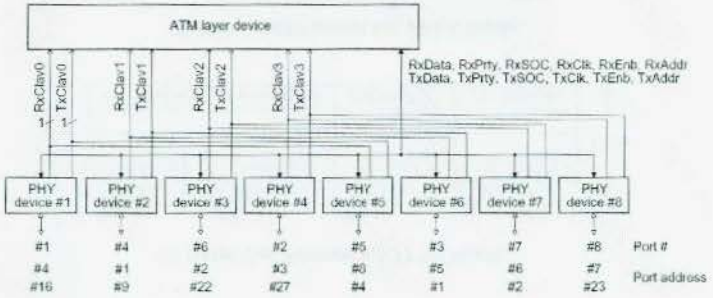


Figure 207: Multiplexed status polling (example 1)

Figure 208 shows an example (transmit direction) for four PHY devices, 31 PHY ports, 8 PHY ports per device and each PHY device with only one status signal per direction. Except for group #7, four PHY devices respond simultaneously to a poll cycle.

4 PHY Devices, 8 PHY Port per Device

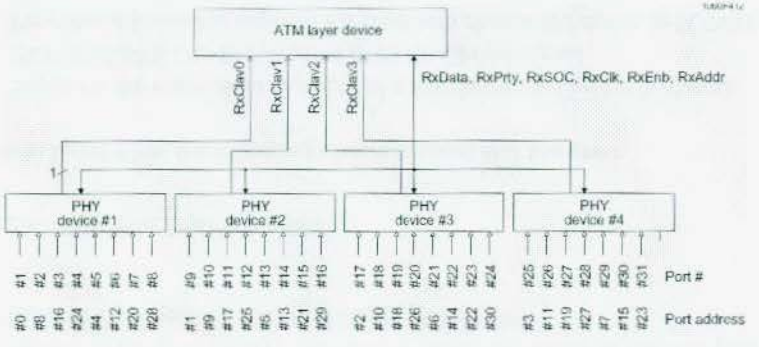


Figure 208: Multiplexed status polling (example 2)

2.4.2.3 *UTOPIA Level 3 Interface*

Utopia level 3 supports single 8-bit, 16-bit or 32-bit wide data path in each direction path operating at frequencies of up to 104 MHz to achieve the aggregate bandwidth required for an OC-48 line interface. The 16-bit data path Is designed to support the bandwidth for a Gigabit line interface. The 8-bit data path supports the bandwidth for an OC-12 line interface.

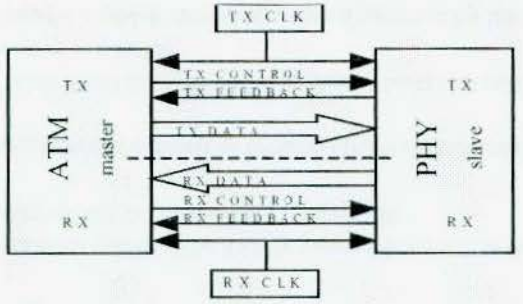


Figure 209: UTOPIA level 3 (L3)

With the introduction of a 32-bit data path, a new cell format is required as the cell size now has to be a multiple of 4 octets. The transfer of 52-octet cells as well as longer cells is allowed. Only the 52- and 56-octet cell formats are

defined in this specification and are as shown in Figure 210. Support for the 52-octet cell format is required (R). Support for the 56-octet cell format is optional (O). Note that in the 52-octet format there is no field for the transport of the HEC octet. However, transport of the HEC is not required, as this is a part of Physical layer functionality. When transferring the HEC in the 56-octet cell format the HEC shall be transferred in the first User Defined Field (UDF1).

52 Octet Cell Format for 32-bit Mode

Bit 31	Bit 23	Bit 15	Bit 0
Header 1	Header 2	Header 3	Header 4
Payload 1	Payload 2	Payload 3	Payload 4
:	:	:	:
:	:	:	:
Payload 45	Payload 46	Payload 47	Payload 48

Time

56 Octet Cell Format for 32-bit Mode

Bit 31	Bit 23	Bit 15	Bit 0
Header 1	Header 2	Header 3	Header 4
UDF 1/HEC	UDF 2	UDF 3	UDF 4
Payload 1	Payload 2	Payload 3	Payload 4
:	:	:	:
:	:	:	:
Payload 45	Payload 46	Payload 47	Payload 48

Time

Figure 210: Cell formats for 32-bit operation mode in UTOPIA L3

The following signals are defined as required (R) for the Transmit interface:

- TxClav[0] (Cell Buffer Available): To indicate that space for at least one cell is available in the PHY transmit cell buffer
- TxEnb* (Transmit Enable): The assertion of TxEnb* is coincident with the start of the cell Transfer
- TxSOC (Transmit Start Of Cell): Active high signal asserted by the ATM layer to indicate the start of cell position
- TxData[7:0]/TxData[15:0]/TxData[31:0]: The data path for Transmit data, from ATM to PHY
- TxClk (Transmit Clock): An input to both the ATM layer device and the PHY device. Used to clock the transmit control signals and data.

The following signal is defined as optional (O) for the Transmit interface:

- TxPrty (Data path parity): The TxPrty parity bit serves as the odd parity bit over TxData[7:0]/ TxData[15:0]/TxData[31:0]

The following signals are defined as required (R) for the Receive interface:

- RxClav[0] (Cell Available): To indicate that at least one cell is available in the PHY receive cell buffer
- RxEnb* (Receive Enable): Active low signal asserted by the ATM layer device to initiate a cell transfer
- RxSOC (Receive Start Of Cell): Active high signal asserted by the PHY layer to indicate the start of cell position. RxSOC is only asserted during the first clock cycle of the data transfer
- RxData[7:0]/RxData[15:0]/RxData[31:0]: The data path for Receive data, from PHY to ATM
- RxClk (Receive Clock): An input to both the ATM layer device and the PHY device. Used to clock the receive control signals and data

The following signal is defined as optional (O) for the Receive interface:

- RxPrty (Data path parity): The RxPrty parity bit serves as the odd parity bit over RxData[7:0]/ RxData[15:0]/RxData[31:0].

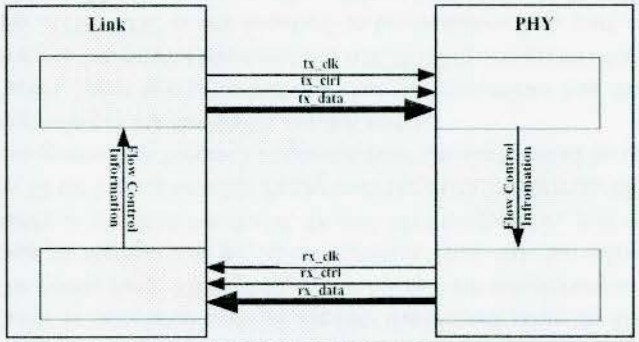
2.4.2.4 UTOPIA Level 4 Interface

Utopia Level 4 has the following characteristics and features:

- Supports the equivalent capacity of a Synchronous Optical Network (SONET/SDH) OC-192/STM-64 (9.95328 Gbit/s) signal
- Provides addressing support for payloads channeled down to SONET STS-1 with addressing support for even deeper channeling
- Minimizes the number of signals required on the interface by moving all control functions in-band
- Supports interconnections across motherboard, daughterboard and backplane interfaces
- Completely symmetrical interface in Tx and Rx directions for easier design and testing (e.g., simple loopback for fault isolation), and wider

applicability

- Simple and efficient flow-control allows handshaking with up to 24 concurrent channels while providing flow control information in one clock cycle (no polling required
- The bus width can be 32, 16 or 8-bits. At least one of these widths must be supported
- The basic interface operates at up to 415 MHz



Transfer Ctrl	Rx/tx_ctrl	Description
DATA	0	ATM header/Packet or cell data
CTRL	1	Control information

Name	No. Pins	Function
tx_clk	2	Source clock from ATM to PHY
tx_ctrl	2	Indicates if tx_data is control or data information
tx_data [N-1:0]	2*N	N bit data bus (where N=32 or 16 or 8)

Name	No. Pins	Function
rx_clk	2	Source clock from PHY to ATM
rx_ctrl	2	Indicates if rx_data is control or data information
rx_data [N-1:0]	2*N	N bit data bus (where N=32 or 16 or 8)

Figure 211: UTOPIA level 4 (L4) specification and signal definition

The 32-bit interface requires 68 pins in each direction for a total of 136 pins full duplex. The 16-bit interface requires 36 pins in each direction for a total of 72 pins full duplex. The 8-bit interface requires 20 pins in each direction for a total of 40 pins full duplex. The symmetry of Utopia Level-4 is illustrated in Figure 211.

The CTRL transfer is used to transmit multi-PHY address information, flow control information and start-of-cell information. Table 24 shows the possible values of rx/tx_data when rx/tx_ctrl = '1'.

- ADR: Each data transfer (both cell and packet data) must be preceded by an ADR word. The ADR word is used to indicate the start of the next cell/packet and the end of the preceding packet. ATM cell data is not required to be followed by an ADR word. The format of the ADR word indicates support for up to 255 PHY channels. The ADR word also has the following properties:
  a) Start of Cell or Packet (SOCP) indicates the start of a cell or packet and End of Packet (EOP) indicates the end of a packet. Start of Packet (SOCP) (with a new address) may be indicated concurrently with EOP.
  b) The SIZE field indicates valid bytes in data transfers (one 32 bit logical word) of between 1 and 4 bytes. This field is valid only with EOP='1' indicating that the previous logical word (32 bits) was the final data transfer of a packet. The number of valid bytes 1 (only the MSB is valid), 2, 3 or 4 are indicated by the SIZE field values of '01', '10', '11' or '00' respectively.
  c) The P bit indicates whether the data being transferred is an ATM cell (logic '0') or a packet (logic '1').
  d) The two incremental congestion fields, CONG[7:0] and REL[7:0], are used for flow control by indicating changes in full conditions of one of the 255 possible ports. The congestion field, CONG[7:0], indicates that the port addressed at CONG[7:0] has just undergone a full condition. The release field, REL[7:0], indicates that port addressed at REL[7:0] has just come out of a full condition. If no congestion change has occurred, a CONG[7:0] or REL[7:0] value of all '1's is sent. Bit 24 of the ADR word may disable the incremental congestion field in cases where backpressure is not needed.
  e) The ADR[7:0] contains the address for the next data transfer. If no data is available and an idle.
- Flow Control (FLC) is to be transferred next, the ADR[7:0] field shall be set to all '1's. An ADR word prior to idle FLC transfers allows end of cell and EOP closure to the last valid cell or packet transfer. When a channel is provisioned for ATM operation, the EOP and SIZE fields are not used and shall be set to zero: the completion of the 52nd payload byte transfer automatically marks the end of the cell.

- FLC: When received, Flow Control words tell the local transmitter (see Figure 2) to stop sending DATA within 40 word transfers (as defined in sections 4.2, 4.3, and 4.4). They may be interspersed among DATA and other CTRL transfers. Section 6 defines the port mapping for this field.
  a) Indications may simultaneously be provided for 24 channels on FULL[23:0].
  b) The block bits, BLK[3:0], select which group of 24 channels the FULL[23:0] signals are reporting, supporting the 255 channels. The use of the reserved bits permits channeling to be increased further. The FLC transfers shall also be used (with none of the full bits asserted if none apply) for idle transfers when no other data or control information needs to be transferred to give faster update of flow control status.
- EXT: Used for extended functions for future standardization. The presently defined uses are for the Parity and Abort Packet words.
  a) Parity is calculated over all the bits transferred since the last parity. The parity field, PRTY[7:0], would be an 8-bit-interleaved odd parity over all the bytes in the 32-bit transfers. Note that the calculation of parity is identical for the 32, 16 and 8-bit modes, (i.e., it is still based on 32-bit logical words). Parity over the rx/tx_ctrl signal, PRTY_C, is also generated. Figure 3 illustrates how the parity shall be calculated beginning at the last parity control word.
  b) Packet Abort is sent to indicate that the transmitter has determined that the last packet from address ABORT[7:0] should be dropped.
  c) The ATM HEC is not required to be transferred as part of Utopia Level 4. However, users who wish to implement it should use this EXT field. The nominal location for this extension would be immediately following the Virtual Path Identifier (VPI)/Virtual Channel Identifier (VCI) data transfer.
- UDF: Reserved for User Defined Fields. The frequency and location of UDF transfers relative toother transfers are not defined by this specification, and may be user-defined.

Name	Rx/Tx data Bit Definitions									Function
	31:30	29	28	27:26	25	24	23:8		7:0	
ADR	00	SOCP	EOP	SIZE	P	0	CONG [7:0]	REL [7:0]	ADR [7:0]	Next PHY address, Incremental Congestion, SOCP, EOP
		SOCP	EOP	SIZE	P	1	*		ADR [7:0]	Next PHY address, SOCP, EOP
FLC	01	00		BLK [3:0]			FULL [23:0]			Flow control, Idle
		01-11		*						Reserved for future definition
EXT	10	0000		*				PRTY_C	PRTY [7:0]	Parity
		0001		*					ABORT [7:0]	Packet Abort
		0010		*					HEC [7:0]	ATM Cell HEC field
		0011-1111		*						Reserved for future definition
UDF	11	**								Reserved for user defined fields

* Reserved for future use and shall be set to zero when transmitted, ignored when received.
** Reserved for users.

Table 24: Format of control information (Rx/Tx_Ctrl = 1)

Differing from previous Utopia specifications where the ATM host is the source of both transmit and receive timing, this specification utilizes a source synchronous timing method as shown in Figure 212. The advantage of using the source synchronous clocking mechanism is that far end data recovery is much more easily achieved. There is no need to account for chip I/O delays or use Phase Lock Loop techniques to ensure clock/data alignment.

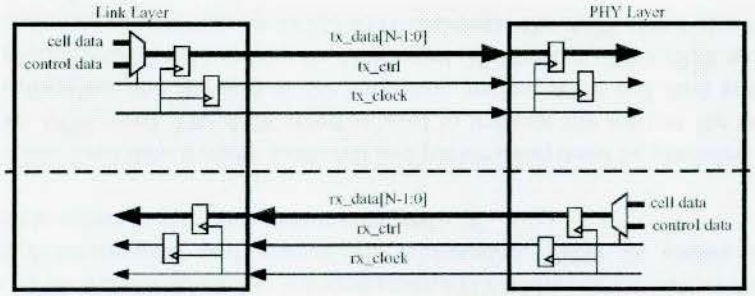


Figure 212: Source synchronous timing in UTOPIA L4

2.5 Fibre Channel (FC) Systems

Fibre Channel has made the biggest impact in the storage arena, in particular, using SCSI as an upper layer protocol. Compared with traditional SCSI, the benefits of mapping the SCSI command set onto Fibre Channel include:

- faster speed,
- more devices can be connected together, and
- larger distance allowed between devices.

Essentially, Fibre Channel, using the Arbitrated Loop topology, has been simply used as a replacement for SCSI. Many companies are shipping SCSI adapter cards for several platforms and operating systems, as well as disk drives and storage devices with Fibre Channel interfaces.

2.5.1 Fibre Channel Layers

Fibre Channel does not follow the OSI reference model, but instead, the protocol has been broken into five layers: FC-0, FC-1, FC-2, FC-3, and FC-4. Each is briefly described below along with the main functions it defines.

FC-0	FC-1	FC-2
Signaling Media specifications Receiver/Transmitter specifications	8B/10B character encoding Link maintenance	Frame format Sequence management Exchange management Flow Control Classes of Service Login/Logout Topologies Segmentation and Reassembly
FC-3	**FC-4**	
Services for multiple ports on one node	Upper Layer Protocol (ULP) mapping	

Table 25: FC layers

FC-4 includes mapping to:

- Small Computer System Interface (SCSI)
- Internet Protocol (IP)
- High Performance Parallel Interface (HIPPI)
- Asynchronous Transfer Mode - Adaption Layer 5 (ATM-AAL5)
- Intelligent Peripheral Interface - 3 (IPI-3) (disk and tape)
- Single Byte Command Code Sets (SBCCS)

FC-0 and FC-1 can be thought of as defining the Physical Layer of the OSI model. FC-2 is similar to what other protocols define as a Media Access Control (MAC) layer, which is typically the lower half of the Data Link layer. Fibre Channel, however, does not define the concept of a MAC. FC-3 is not really a layer at all. It is still a largely undefined set of services for devices having more than one port. An example is striping, where data is transmitted out of all ports at the same time in order to increase bandwidth. FC-4 defines how other well-known higher layer protocols are mapped onto and transmitted over Fibre Channel. Thus, one can roughly think of the Fibre Channel layers defining up through the Transport layer of the OSI model.

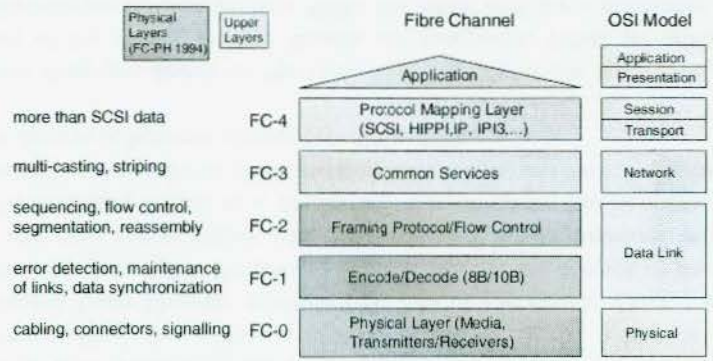


Figure 213: Fibre Channel architecture

2.5.2 Fibre Channel Topologies

Fibre Channel defines three topologies, namely Point-to-Point, Arbitrated Loop, and Fabric. Each of these is described below.

2.5.2.1 *Point-to-Point*

A Point-to-Point topology is the simplest of the three. It consists of two and only two Fibre Channel devices connected directly together. The transmit fibre of one device goes to the receive fibre of the other device, and vice versa. There is no sharing of the media, which allows the devices to enjoy the total bandwidth of the link. A simple link initialization is required of the two devices before communication can begin.

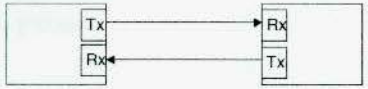


Figure 214: Point-to-point topology

2.5.2.2 *Arbitrated Loop*

Arbitrated Loop has become the most dominant Fibre Channel topology, but it is also the most complex. It's a cost-effective way of connecting up to 127 ports in a single network without the need of a Fabric switch. Unlike the other two topologies, the media is shared among the devices, limiting each device's access. Not all devices are required to operate on an Arbitrated Loop; the added functionality is optional. Thus, for a Loop to operate, all devices must be Loop devices.

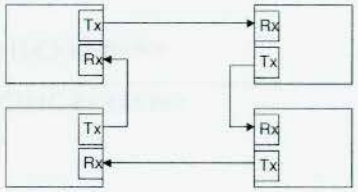


Figure 215: Arbitrated loop topology

Arbitrated Loop is not a token-passing scheme. When a device is ready to transmit data, it first must arbitrate and gain control of the Loop. It does this by transmitting the Arbitrate (ARBx) Primitive Signal, where x = the Arbitrated Loop Physical Address (AL_PA) of the device. Once a device receives its own ARBx Primitive Signal, it has gained control of the Loop and can now communicate with other devices by transmitting an Open (OPN) Primitive Signal to a destination device. Once this happens, there essentially exists point-to-point communication between the two devices. All other devices in between simply repeat the data.

If more than one device on the Loop is arbitrating at the same time, the x values of the ARB Primitive Signals are compared. When an arbitrating device receives another device's ARBx, the ARBx with the numerically lower AL_PA is forwarded, while the ARBx with the numerically higher AL_PA is blocked. Thus, the device with the lower AL_PA will gain control of the Loop first. Once that device relinquishes control of the Loop, the other device can have a chance.

Unlike token-passing schemes, there is no limit on how long a device may retain control of the Loop. This demonstrates the "channel" aspect of Fibre Channel. There is, however, an Access Fairness Algorithm, which prohibits a device from arbitrating again until all other devices have had a chance to arbitrate. The catch is that the Access Fairness Algorithm is optional.

2.5.2.3 *Fabric*

The Fabric topology is used to connect many (224) devices in a cross-point switched configuration. The benefit of this topology is that many devices can communicate at the same time; the media is not shared. Of course, it also requires the purchase of a switch.

When the N_Ports log into the Fabric, the Fabric will assign Native Address Identifiers (S_ID). Other functions of the Fabric include a multicast server, broadcast server, alias server, quality of service facilitator, and directory server. Some Fabrics have FL_Ports, allowing Arbitrated Loops to be connected to the Fabric.

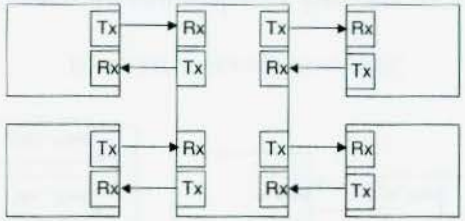


Figure 216: FC fabric

Initialization

Before the Loop is usable, it must be initialized so that each port obtains an Arbitrated Loop Physical Address (AL_PA), a dynamically assigned value by which the ports communicate. It maps to the lowest byte of the Native Address Identifiers (D_ID and SI_ID). Although the AL_PA is a byte, only 127 values are valid (neutral running disparity). If more than 127 devices are present on the Loop, some will not be able to select an AL_PA and will be forced into non-participating mode. Without getting into all the details of Loop initialization, the basics are as follows.

The LIP Primitive Sequence begins the process. LIP is transmitted by an L_Port after it powers on, or when it detects Loop Failure (loss of synchronization at its receiver). The LIP will propagate around the Loop, triggering all other L_Ports to transmit LIP as well. At this point, the Loop is not usable.

Frame	Reason	Priority/Order Transmitted
LIFA (Loop Initialization Fabric Assigned)	a certain AL_PA was assigned by the Fabric	1
LIPA (Loop Initialization Previously Acquired)	before this initialization, the L_Port had a valid AL_PA	2
LIHA (Loop Initialization Hard Assigned)	the L_Port has a certain AL_PA it tries to claim	3
LISA (Loop Initialization Soft Assigned)	the L_Port claims the first available AL_PA that is left	4

Table 26: Arbitrated loop initialization

The next major step is to select a Loop master that will control the process of AL_PA selection. This is done by the L_Ports constantly transmitting Loop Initialization Select Master (LISM) frames. The process is designed so that if an Fabric is present, it will become Loop master (by receiving back its own LISM frames), otherwise, the port with the numerically lowest Port Name will win. All other L_Ports propagate the higher priority LISM frames.

The third step is to allow the L_Ports to select an AL_PA. The concept of an AL_PA bitmap is used; where each L_Port selects (and sets) a single bit in the bitmap of a frame originated by the Loop master and repeats the frame back on the Loop. There are 127 available bits, corresponding to the 127 valid AL_PAs. This process is done using four frames, breaking the selection down according to priority.

So for example, if an L_Port had a valid AL_PA before the Loop began this initialization, it will attempt to reclaim this previously acquired value by looking for that bit to be available in the LIPA frame. If it is, it will set the bit and repeat the frame. If it is not available (already been claimed), the L_Port will wait for the LISA frame to come around and claim one there.

Once the LISA frame has come back to the Loop master, all L_Ports (hopefully) have selected an AL_PA. Two additional frames may be sent by the Loop master, but only if all L_Ports on the Loop support them. The first is the Loop Initialization Report Position (LIRP). As the frame traverses the Loop, each port adds its AL_PA to the end of a list. When done, the relative positions of all L_Ports are known. Finally, the Loop Initialization Loop Position (LILP) frame is transmitted, which simply allows all L_Ports to look at the finished list.

Whether or not LIRP and LILP are used, the Loop master transmits the CLS (Close) Primitive Signal to let each port know that the process has finished. At this point, the Loop has finished initializing and ready to be used.

Fibre Channel Ports

All equipment that is connected to a Fibre Channel network must contain at least one Fibre Channel port. The ports are able to send or receive data under the Fibre Channel Protocol. Each port type has its own characteristics, and is required to connect to a limited set of port types on the other end of

the connection to create a valid Fibre Channel configuration. The available ports are:

- N_Port: N (Node) ports are the simplest ports, and are implemented on all devices like servers and storage units. An N_Port may only participate in a point-to-point connection with another N_port or with an F_port on a switch.
- NL_Port: NL (NodeLoop) ports are N_Ports with the additional functionality of being able to participate on an arbitrated loop.
- F_Port: F (Fabric) ports are ports used on a Fibre Channel switch to connect it to N_Ports on nodes.
- FL_Port: FL (FabricLoop) ports on switches allow a switch to participate in an arbitrated loop meaning that all devices on the loop will be able to access everything that the switch spans.
- E_Port: E (Expansion) ports provide a means for connecting a switch to another switch. The existence of E_Ports is the glue for larger fabric configurations of switches.
- G_Port: G (Generic) ports are ports on a switch that can act either as an E_Port, FL_Port, or F_Port, depending on which port it is connected to.

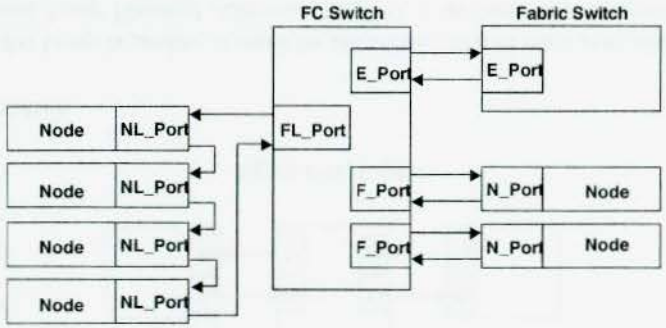


Figure 217: Port assignments in FC

The ports can be grouped a couple of ways. First, any port on a node device, e.g., a disk, a PC, is an N_Port as compared with a port on a Fabric, which is an F_Port. At the same time, any port that happens to have Arbitrated Loop capabilities, it is called an L_Port. Combining these groupings, we also have NL_Ports and FL_Ports.

Ports that connect switches together are called E_Ports. They generally do not need to follow Fibre Channel rules and protocols. A port that can act as either an E_Port or an F_Port is called a G_Port. Finally, a G_Port with Loop capabilities is a GL_Port.

2.5.3 Media Options

Speed

Fibre Channel offers a very wide range of media speeds. One of the goals of FC is to allow HIPPI to map to it. HIPPI is 100 MByte/s, 200 MByte/s and 800 MByte/s technologies, thus Fibre Channel's primary data rate allows for data to travel 100 MByte/s. After factoring in 8B/10B encoding, frame headers, and other overhead, the transmission speed is 1063 Mbit/s. This speed is referred to as full speed. There also exists half speed, quarter speed, and eighth speed. In addition, double and quadruple speeds are defined as Table 27 illustrates. By far, the most common speed is full speed, with some quadruple speed devices also in existence. Note that there is a new standard adopted by the American National Standard Institute (ANSI) in 2004 that specify 10 Gigabit transmission rate for Fibre Channel. This standard uses similar transmission characteristics as those used in 10 GbE (8B/10B and 64B/66B coding schemes, scrambling, definitions of XGMII and XSBI...), but at a line rate of 10.2 Gbit/s.

9 Micrometer Single-Mode Fiber				
Speed (MByte/s)	Rate (Mbit/s)	Distance	Laser Type	
100	1062.5	up to 10 km	Longwave	
50	531.2	up to 10 km	Longwave	
25	265.6	up to 10 km	Longwave	
50 Micrometer Multi-Mode Fiber				
Speed (MByte/s)	Rate (Mbit/s)	Distance	Laser Type	Optical
100	1062.5	up to 0.5 km	Shortwave	
50	531.2	up to 1.5 km	Shortwave	
25	265.6	up to 2 km	Shortwave	
25	265.6	up to 2 km	Longwave LED	
12.5	132.8	up to 10 km	Longwave LED	
Shielded Twisted Pair				
Speed (MByte/s)	Rate (Mbit/s)	Distance	Logic Levels	Electrical
25	265.6	up to 25 m	ECL	
12.5	132.8	up to 35 m	ECL	

Table 27: Media types and speed options

Media

Despite the name, Fibre Channel can run over both copper and fiber media. Speeds up to 100 MByte/s can run on both copper and fiber; 200 MByte/s and 400 MByte/s require fiber media.

For copper, the following cable types are used: video cable, miniature cable, and shielded twisted pair. The most common by far is shielded twisted pair, using a DB-9 connector. For fiber, the choices are: 62.5 µm multi-mode, 50 µm multi-mode, and single-mode. The SC connector is used. Also for fiber, both long wave and short wave lasers can be used. Short wave seems to be most popular now. The short wave transmitters have the option to implement the open fibre control (OFC). This is a scheme designed to prevent disconnected fibers from continuously transmitting light, as a safety precaution. There is a simple protocol where each transmitter on a link periodically transmits short pulses of light to the other receiver. When the receivers detect this, the transmitters can operate normally. Most devices however, are non-OFC.

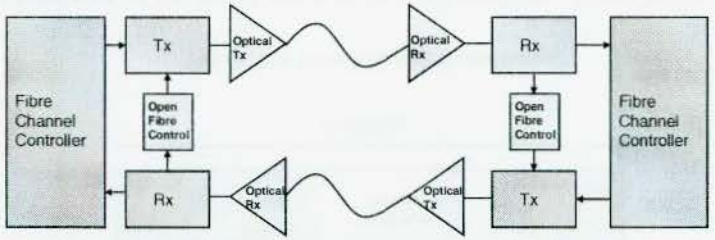


Figure 218: Open fiber control system (OFC)

2.5.4 Transmission Hierarchy

The easiest way to understand the methods by which information is transmitted over Fibre Channel is by looking at the problem in the form of a hierarchy. At the lowest level, Fibre Channel uses the IBM 8B/10B encoding scheme. Basically, every byte of data that is to be transmitted is first converted into a 10-bit value called a Transmission Character. Using this encoding scheme has the following benefits:

- Improvement of the transmission characteristics of information to be transferred,
- Provides enough transitions to make clock recovery possible at the receiver,
- Improves the detection of single and multiple bit errors, and
- Some Transmission Characters contain a unique bit pattern (comma) to aid in achieving word alignment.

Fibre Channel defines a "1" to simply be the state with more optical power (for optical links), or the state where the "+" pin is more positive than the "-" pin (in the case of copper). The 8B/10B encoding uses the idea of running disparity, which is concerned with the number of 1s and 0s in each Transmission Character. Running disparity is evaluated after the first 6 bits of each Transmission Character and again after the last 4 bits and can be either positive (more 1s than 0s) or negative (more 0s than 1s). It is desirable to try and equalize the number of 1s and 0s over time. Thus, every byte to be transmitted is encoded into one of two 10-bit representations depending on the current running disparity.
As stated, every byte to be transmitted is first converted into a 10-bit Transmission Character. But there are many more possible 10-bit Transmission Characters than are needed to map to particular bytes. Most of the remaining 10-bit encodings are not defined, and only one is used. This is the special K28.5 Transmission Character, which contains the "comma", a 7-bit string that cannot occur in any Data Transmission Character. Because of this, the K28.5 is used as a special control character.

Transmission Word

All information in Fibre Channel is transmitted in groups of four Transmission Characters called Transmission Words. Some Transmission Words have K28.5 as the first Transmission Character. These are special Transmission Characters called Ordered Sets. Some Ordered Sets mark the beginning and end of frames (frame delimiters). Others convey information in between frames in the form of Primitive Signals (a single Ordered Set) and Primitive Sequences (a stream of the same Ordered Set). Examples of Ordered Sets are: Start of Frame (SOF), End of Frame (EOF), Idle, Receiver_Ready (R_RDY), Loop Initialization Primitive (LIP), Arbitrate (ARB), Open (OPN), Close (CLS), and several others.

Frame

Fibre Channel defines a variable length frame consisting of 36 bytes of overhead and up to 2112 bytes of payload for a total maximum size of 2148 bytes.

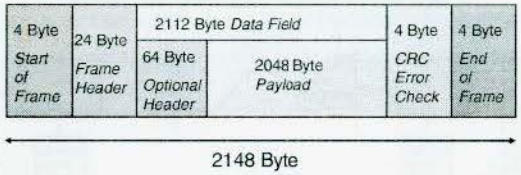


Figure 219: Fibre Channel frame format

The total size of the frame must be an even multiple of four bytes so that partial Transmission Words are not sent. Between 0 and 3 pad bytes are appended to the end of the payload to satisfy this rule. A Start of Frame (SOF) delimiter and End of Frame (EOF) delimiter mark the beginning and end of each Fibre Channel frame. The Cyclic Redundancy Check (CRC) is the same 32-bit CRC used in FDDI.

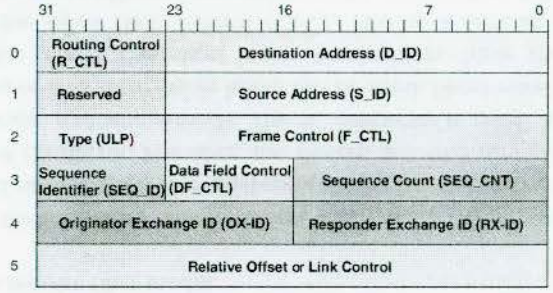


Figure 220: Fibre Channel frame header

Sequence

A Fibre Channel Sequence is a series of one or more related frames transmitted unidirectional from one port to another. All frames must be part of a Sequence. Frames within the same Sequence have the same SEQ_ID field in the header. The SEQ_CNT field identifies individual frames within a Sequence. For each frame transmitted in a Sequence, SEQ_CNT is

incremented by 1. This provides a means for the recipient to arrange the frames in the order in which they were transmitted and to verify that all expected frames have been received. Multiple Sequences to multiple ports may be active at a time.

Exchange

A Fibre Channel Exchange is a series or one or more non-concurrent Sequences between two ports. The Sequences may be in either direction. All Sequences (and therefore all frames) must be part of an Exchange. The originator of the Exchange assigns the OX_ID field. The responder assigns the RX_ID field. As another perspective, one can use the following analogy:

- frame ~ word
- sequence ~ sentence
- exchange ~ conversation

Of course, one main difference is that a Fibre Channel device can "speak" more than one sentence and hold more than one "conversation" at a time.

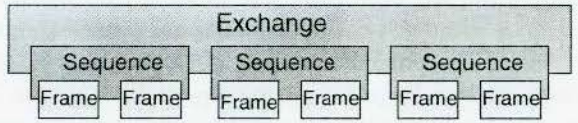


Figure 221: Fibre Channel framing hierarchy

2.5.5 Flow Control

The concept of flow control deals with the problem where a device receives frames faster than it can process them. When this happens, the result is that the device is forced to drop some of the frames. Fibre Channel has a built-in flow control solution to this problem.

The idea is simple enough. A device can transmit frames to another device only when the other device is ready to accept them. Before the devices can send data to each other, they must login to each other. One of the things accomplished in login is establishing credit. Credit refers to the number of

frames a device can receive at a time. This value is exchanged with another device during login, so each knows how many frames the other can receive. After enough frames have been transmitted and credit runs out, no more frames can be transmitted until the destination device indicates it has processed one or more frames and is ready to receive new ones. Thus, no device should ever be overrun with frames. Fibre Channel uses two types of flow control, buffer-to-buffer and end-to-end.

Buffer-to-Buffer Flow Control

This type of flow control deals only with the link between an N_Port and an F_Port or between two N_Ports. Both ports on the link exchange values of how many frames it is willing to receive at a time from the other port. This value becomes the other port's BB_Credit value and remains constant as long as the ports are logged in. For example, when ports A and B log into each other, A may report that it is willing to handle 4 frames from B; B might report that it will accept 8 frames from A. Thus, B's BB_Credit is set to 4, and A's is set to 8.

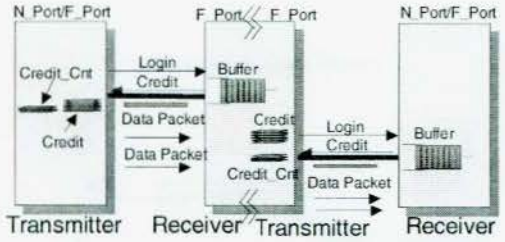


Figure 222: Buffer-to-buffer flow control

Each port also keeps track of BB_Credit_CNT, which is initialized to 0. For each frame transmitted, BB_Credit_CNT is incremented by 1. The value is decremented by 1 for each R_RDY Primitive Signal received from the other port. Transmission of an R_RDY indicates the port has processed a frame, freed a receive buffer, and is ready for one more. If BB_Credit_CNT reaches BB_Credit, the port cannot transmit another frame until it receives an R_RDY.

End-to-End

End-to-End flow control is not concerned with individual links, but rather the source and destination N_Ports. The concept is very similar to buffer-to-buffer flow control. When the two N_Ports log into each other, they report how many receive buffers are available for the other port. This value becomes EE_Credit. EE_Credit_CNT is set to 0 after login and increments by 1 for each frame transmitted to the other port. It is decremented upon reception of an ACK Link Control frame from that port. ACK frames can indicate the port has received and processed 1 frame, N frames, or an entire Sequence of frames.

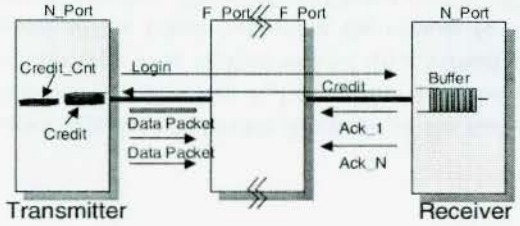


Figure 223: End-to-end flow control

2.5.6 Classes of Service

Fibre Channel defines several communication strategies called Classes of service. The Class used greatly depends on the type of data to be transmitted. The major difference between among the Classes is the types of flow control used. If two N_Ports are to communicate or if an N_Port is to successfully log into a Fabric, there must be at least 1 common Class of service supported between them, since Sequences and Exchanges must take place using a single Class of service. This information is exchanged during Fabric Login and N_Port Login.

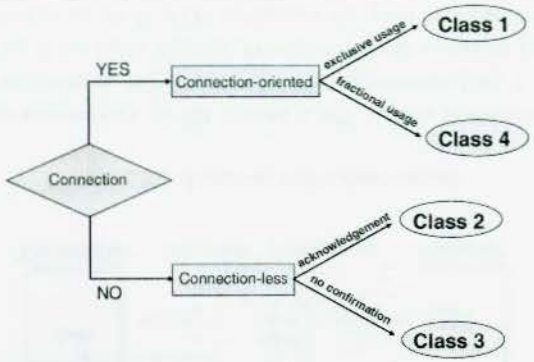


Figure 224: Service classes

Class 1

In Class 1, a dedicated connection is established between two N_Ports. Once established, the two N_Ports may communicate using the full bandwidth of the connection; no other network traffic affects this communication. Due to this, frames are guaranteed to arrive in the order in which they were transmitted. In addition, the media speeds must be the same for all links which make up the dedicated connection. Because of the nature of the dedicated connection, there is no need for buffer-to-buffer flow control; the Fabric does not need to buffer the frames as they are routed. Thus, only end-to-end flow control is used in Class 1. Class 1 would be used when the data needs to be continuous and time critical, such as voice or video.

Intermix - Intermix is an option of Class 1 whereby Class 2 and Class 3 frames may be transmitted at times when Class 1 frames are not being transmitted. The Class 2 and Class 3 frames may or may not be destined to the same N_Port as the Class 1 frames. Both N_Ports as well as the Fabric must support Intermix for it to be used.

Class 2

Class 2 is referred to as multiplex due to the fact that it is a connectionless Class of service with notification of delivery and non-delivery of frames. Since no dedicated connection needs to be established, a port can transmit frames to and receive frames from more than one N_Port. As a result, the

N_Ports share the bandwidth of the links with other network traffic. Frames are not guaranteed to arrive in the order in which they were transmitted, except in the point-to-point or Loop topologies. Also, the media speeds may vary for different links which make up the path. Both buffer-to-buffer and end-to-end flow control are used in Class 2. Class 2 is more like typical LAN traffic, such as IP or FTP, where the order and timeliness of delivery is not so important.

Class 3

Class 3 is very similar to Class 2. The only exception is that it only uses buffer-to-buffer flow control. It is referred to a datagram service. Class 3 would be used when order and timeliness is not so important, and when the ULP itself handles lost frames efficiently. Class 3 is the choice for SCSI.

Class 4

Class 4 provides fractional bandwidth allocation of the resources of a path through a Fabric that connects two N_Ports. Class 4 can be used only with the pure Fabric topology. One N_Port will set up a Virtual Circuit (VC) by sending a request to the Fabric indicating the remote N_Port as well as quality of service parameters. The resulting Class 4 circuit will consist of two unidirectional VCs between the two N_Ports. The VCs need not be the same speed.

Like a Class 1 dedicated connection, Class 4 circuits will guarantee that frames arrive in the order they were transmitted and will provide acknowledgement of delivered frames (Class 4 end-to-end credit). The main difference is that an N_Port may have more than one Class 4 circuit, possibly with more than one other N_Port at the same time. In a Class 1 connection, all resources are dedicated to the two N_Ports. In Class 4, the resources are divided up into potentially many circuits. The Fabric regulates traffic and manages buffer-to-buffer flow control for each VC separately using the FC_RDY Primitive Signal. Intermixing of Class 2 and 3 frames is mandatory for devices supporting Class 4.

Class 5

The idea for Class 5 involved isochronous, just-in-time service. However, it is still undefined, and possibly scrapped altogether. It is not mentioned in any of the FC-PH documents.

Class 6

Class 6 provides support for multicast service through a Fabric. Basically, a device wishing to transmit frames to more than one N_Port at a time sets up a Class 1 dedicated connection with the multicast server within the Fabric at the well-known address of hex'FFFFF5'. The multicast server sets up individual dedicated connections between the original N_Port and all the destination N_Ports. The multicast server is responsible for replicating and forwarding the frame to all other N_Ports in the multicast group. N_Ports become members of a multicast group by registering with the Alias Server at the well-know address of hex'FFFFF8'. The Class 6 is very similar to Class 1; Class 6 SOF delimiters are the same as used in Class 1. Also, end-to end flow control is used between the N_Ports and the multicast server.

2.5.7 Addressing

Unlike many LAN technologies that use a fixed six-byte Media Access Control (MAC) address, Fibre Channel uses a three byte address identifier, which is dynamically assigned during Login. N_Ports transmit frames from their own Source_ID (S_ID) to a Destination_ID (D_ID). Addresses in the range of hex'FFFFF0' to hex'FFFFFE' are special, well-known addresses uses for such things as the Fabric, Alias Server, or the Multicast Server. Before Fabric Login, the N_Port's S_ID is undefined: hex'000000'. Hex'FFFFFF' is reserved for broadcast. In a point-to-point topology, Fabric Login will fail of course, and the two ports will simply chose two unique addresses.

Arbitrated Loop devices still use the three byte address identifiers, but also use an Arbitrated Loop Physical Address (AL_PA). AL_PAs are one byte values dynamically assigned each time the Loop is initialized. Once the Loop is initialized and (hopefully) each L_Port has selected an AL_PA, public NL_Ports will attempt Fabric Login. If there is an FL_Port, the Fabric will assign the upper two bytes of the NL_Port's address identifier and

usually allow the low byte to be the NL_Port's AL_PA. (If not, the Loop will need to be re-initialized so the NL_Port can select the Fabric assigned AL_PA). If no Fabric exists or if an NL_Port is a private NL_Port (does not login with the Fabric), the upper two bytes of the address identifier will remain '0000', and the lower byte will simply be the NL_Port's AL_PA.

But there still needs to be a way of uniquely identifying a port - even for much of the above initialization to take place. This is accomplished using Name_Identifiers, a fixed 64-bit value. Name_Identifiers are used to uniquely identify nodes (Node_Name), a Port (Port_Name), and a Fabric (Fabric_Name). Name Identifiers are not used to route frames, but are used in mapping to upper layer protocols.

2.5.8 Login

Fibre Channel defines two types of login procedures, Fabric and N_Port. With the exception of private NL_Ports, all other node ports must attempt to log in with the Fabric. This is typically done right after the link or the Loop has been initialized. It consists of the node port transmitting a Fabric Login (FLOGI) frame to the well-known Fabric address hex'FFFFFE'. The normal response is an Accept (ACC) frame from the Fabric back to the node port. Fabric Login accomplishes the following operations:

- It determines the presence or absence of a Fabric.
- If a Fabric is present, it provides a specific set of operating characteristics associated with the entire Fabric, including which Classes of service are supported.
- If a Fabric is present, it shall optionally assign or shall confirm the native N_Port Identifier of the N_Port that initiated the Login.
- If a Fabric is not present, an ACC from an N_Port indicates that the requesting N_Port is attached in a point-to-point topology.
- If a Fabric is present, it initializes the buffer-to-buffer credit.

Before a node port can communicate with another node port, it must first perform N_Port Login with that node port. Similar to Fabric Login, the process entails transmitting a PLOGI frame to the destination node port. Again, the normal response is an ACC frame. N_Port Login accomplishes the following operations:

- It provides a specific set of operating characteristics associated with the destination N_Port, including which Classes of service are supported.
- Initializes the destination end-to-end credit.
- In point-to-point topology, buffer-to-buffer credit is initialized.

Both Fabric Login and N_Port Login are intended to be long-lived. Once logged in, a device can stay logged in indefinitely, even if it has no further data to transmit at that time.

2.5.9 FC Components

2.5.9.1 FC Hubs

Fibre Channel hubs provide a cheap and easy solution to sharing storage and servers for a storage area network (SAN). The hubs implement an arbitrated loop as a physical star, offering bypass circuitry to avoid failing units and unconnected ports to bring down the loop. The big drawback to hubs is that the devices connected to it share the bandwidth. Typical hubs come with 6 to 12 ports, although there are hubs with as many as 32 ports that can be used if there are multiple devices that need to be connected each having relatively little data traffic. The connectors on the hubs are either fixed for really low cost solutions or through GBICs (GigaBit Interface Converter) for more flexibility.

2.5.9.2 Switches & Directors

Switches offer the most flexible solution for implementing SANs Switches are able to route data between two ports on the same port directly, and have the ability to redirect traffic to other paths within the switch if necessary. Some switches implement a fully connected backbone between the ports, while the cheaper and larger ones offer slightly lesser connectivity. The port count range from 8 and 32 ports on regular switches to 16 and 224 ports on enterprise level director switches.

The switches can be cascaded through fabric ports to form large fabrics, which ensure easy expansion when the amount of storage needs to be increased, or additional servers need to be added. Most switches also allow

multiple links to be used for interconnecting switches, meaning that the traffic can be load balanced and congestion can be avoided.

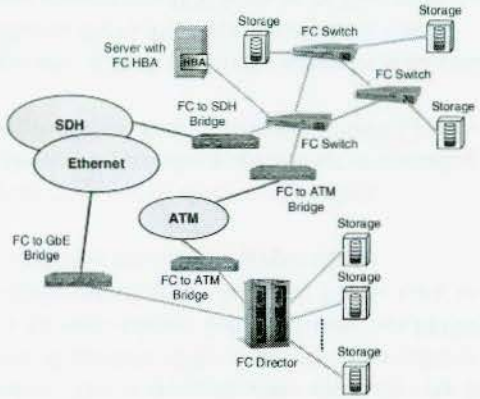


Figure 225: Fibre Channel components

2.5.9.3 Host Bus Adapters (HBAs)

A host-bus-adapter is an intelligent I/O processor that plugs into a host, providing the host with a Fibre Channel port. HBAs can process block-level I/O without utilizing the host's CPU. A server may utilize as few as two HBAs (for redundancy purposes) to dozens of them, depending on the size of the server. The vendors are also increasing dual port HBAs, to increase the number of ports available for a given server.

2.5.9.4 Bridges

A standard component in many SANs is a Fibre Channel bridge, which allows devices from other network technologies to coexist with Fibre Channel devices on the SAN. Since serverless backup is one of the most important applications enabled by SANs, and only a limited number of Fibre Channel tape drives exist, Fibre Channel to SCSI bridges are common. The bridges allow legacy SCSI tape drives and other devices to participate in a Fibre Channel Storage Area Network, meaning that old devices can also be included in the SAN. Other types of bridges also exist, including FC to ATM, FC to Ethernet, and FC to SONET/SDH.

List of Figures

Figure 1: Interconnecting Technologies 5
Figure 2: Block diagram of a PCI bus system 7
Figure 3: PCI in a communication system 9
Figure 4: The PCI read transfer burst 9
Figure 5: The PCI write transfer burst 11
Figure 6: Configuration address space 13
Figure 7: Accessing - address space configuration 14
Figure 8: Base address registers 14
Figure 9: PCI Express 17
Figure 10: PCI Express architectures 17
Figure 11: Layered PCI architecture 18
Figure 12: PCI Express device layers 19
Figure 13: PCI Express bus 19
Figure 14: PCI Express lanes 20
Figure 15: Physical layer of PCI Express 20
Figure 16: Transaction layer packets 21
Figure 17: PCI Express in server platforms 21
Figure 18: PCI Express in client platforms 22
Figure 19: PCI Express in communication platforms 23
Figure 20: PCI Express form factors 23
Figure 21: Small computer system interface (SCSI) basics 24
Figure 22: SCSI History 26
Figure 23: SCSI-3 27
Figure 24: Serial attached SCSI (SAS) roadmap 29
Figure 25: Operation of the SCSI bus 30
Figure 26: SCSI bus phases 31
Figure 27: SCSI commands 33
Figure 28: A maximum SAS system configuration 37
Figure 29: Protocol layers in a SAS device 38

Figure 30: Typical serial SCSI (SAS) configuration 39
Figure 31: Serial attached SCSI (SAS) applications 39
Figure 32: Internet SCSI (iSCSI) 40
Figure 33: iSCSI adapter implementation options 41
Figure 34: SCSI in storage area networks (SANs) 42
Figure 35: HyperTransport connections 43
Figure 36: HyperTransport device configurations 44
Figure 37: 8-bit HyperTransport link 45
Figure 38: 16-bit HyperTransport link 45
Figure 39: HyperTransport electrical specification 46
Figure 40: HyperTransport packet format 47
Figure 41: HyperTransport reads and writes 47
Figure 42: Priority request interleaving 48
Figure 43: HyperTransport control packet 49
Figure 44: Read response packet format 50
Figure 45: Common HyperTransport command formats 51
Figure 46: Virtual channel packet format 51
Figure 47: Mapping of user data in HyperTrasnport packets 52
Figure 48: Host reflected routing 52
Figure 49: Device-to-device routing 53
Figure 50: Multiprocessor system application 53
Figure 51: Switched networking application 54
Figure 52: HyperTransport high performance cluster application 55
Figure 53: RapidIO interconnect architecture 56
Figure 54: RapidIO specification 56
Figure 55: RapidIO 8/16 LP-LVDS physical layer 57
Figure 56: Operation sequence 58
Figure 57: RapidIO packet formats 59
Figure 58: Cotrol symbols 61
Figure 59: Error management 61
Figure 60: Maintenance 62
Figure 61: RapidIO packet overhead axamples 64
Figure 62: InfiniBand topologies and components 65
Figure 63: InfiniBand in enterprise storage 65
Figure 64: An InfiniBand system area network 66
Figure 65: InfiniBand architecture 67
Figure 66: InfiniBand communication stack 67
Figure 67: Channel adapter 68
Figure 68: Format of the global identifier (GID) 69

Figure 69: InfiniBand packet format 70
Figure 70: Positioning of common switch interface (CSIX) 71
Figure 71: Fabric - point of concentration 72
Figure 72: Common switch interface level 1 (CSIX-L1) implementation 73
Figure 73: CSIX-L1 32-bit interface 74
Figure 74: CSIX-L1 64-bit interface 74
Figure 75: CSIX-L1 64-bit interface signals 75
Figure 76: CSIX-L1 128-bit interface 76
Figure 77: CFrame structure 77
Figure 78: CFrame sequence example 78
Figure 79: Vertical parity 78
Figure 80: Pause and resume in common switch interface (CSIX) 78
Figure 81: Electronic backplanes 81
Figure 82: Switched backplane solutions 83
Figure 83: An example of high-speed Ethernet backplane implementation 84
Figure 84: Box-to-box optical modules 85
Figure 85: Optical interconnects 86
Figure 86: Arrays of vertical cavity surface emitting lasers (VCSELs) 88
Figure 87: Optical backplanes 88
Figure 88: Microchannel interconnect (a- schematic, b- implementation) 89
Figure 89: An example of free-space optical interconnects 90
Figure 90: An example – PAROLI optical backplane 90
Figure 91: PAROLI transceiver module 91
Figure 92: PAROLI backplane connection 91
Figure 93: SONET/SDH networks 94
Figure 94: SONET/SDH network elements 95
Figure 95: SDH transmission path 96
Figure 96: Format of SDH frame 96
Figure 97: Mapping in SDH 98
Figure 98: Regenerator and multiplexer section overheads (SOHs) 98
Figure 99: Path overhead (POH) 99
Figure 100: Bit error monitoring in SDH 99
Figure 101: Byte wise multiplexing of N STM signals 100
Figure 102: SONET/SDH interfaces 101
Figure 103: Electrical Interfaces developed by the OIF 102
Figure 104: SPI-3 Interface 104
Figure 105: SPI-4.1 Interface 105
Figure 106: SPI 4.2 Interface 106
Figure 107: SPI-5 Interface 109

Figure 108: SFI-4.1 Interface 111
Figure 109: SFI-4.2 Interface 112
Figure 110: SFI-5 interface 114
Figure 111: Common electrical interface (CEI) 115
Figure 112: Signal diagram of the common electrical interface (CEI) 116
Figure 113: CEI jitter 117
Figure 114: CEI eye mask 117
Figure 115: CEI wander and skew 118
Figure 116: Reference diagram of TDM fabric interface (TFI) 119
Figure 117: TFI-5 layered approach 120
Figure 118: TFI-5 generic frame format 121
Figure 119: TFI-5 Layers Example (an STS-1 cross-connect) 122
Figure 120: SONET/SDH framer 123
Figure 121: An example – TranSwitch PHAST-12E 123
Figure 122: Framer analysis 124
Figure 123: Line interface 125
Figure 124: ATM over SONET/SDH 126
Figure 125: Finite state machine (FSM) sequential depth 126
Figure 126: ATM cell processing 127
Figure 127: An example – Agere MARS2G5 P-Pro (SONET/SDH framer). 128
Figure 128: An example – Intel WG4500 (SONET/SDH framer) 130
Figure 129: An example – Intel WG1510 (SONET/SDH ADM) 131
Figure 130: SPI-4.2 interface overview 132
Figure 131: Switch design with SPI-4 interface 133
Figure 132: SPI-4 payload mapping 134
Figure 133: FPGA internal structure (Xilinx Virtex II) 134
Figure 134: SPI-4 transmit core overview 135
Figure 135: Transmit core data channel structure 136
Figure 136: Sequence logic state machine 138
Figure 137: Implemented DIP-4 calculation algorithm 139
Figure 138: Transmit core status channel structure 139
Figure 139: Flow control feedback loop 140
Figure 140: General status channel transmission scheme 141
Figure 141: Flow control state machine 142
Figure 142: Moving events across domains 143
Figure 143: Serializer 143
Figure 144: Basic 8:1 multiplexer 144
Figure 145: Critical path timing 144
Figure 146: Serializer scalability 145

Figure 147: Placement of the serializer 145
Figure 148: Multiplexer assembly 147
Figure 149: Optimized serializer scalability 147
Figure 150: Ethernet (IEEE 802.3) 149
Figure 151: Cabling in Ethernet 151
Figure 152: Manchester coding 154
Figure 153: Non-return to zero – inverted on ones (NRZ-I) coding 155
Figure 154: Multilevel transmission encoding – 3 levels (MLT-3) 155
Figure 155: Two dimensional pulse-amplitude modulation (PAM-5x5) 157
Figure 156: Four dimensional PAM-5 code 158
Figure 157: 1000 BASE-T cabling issues 159
Figure 158: 64B/66B code overview 162
Figure 159: Building frames from 10 GbE reconciliation sublayer (RC) symbols 163
Figure 160: Auto-negotiation process (normal and fast link pulses) 164
Figure 161: Specification of Gigabit Ethernet physical layer 167
Figure 162: An example of 10/100 Mbit/s PHY chip 168
Figure 163: An example of GbE PHY chip 168
Figure 164: Block diagram of the 10GBASE-R physical coding sublayer 169
Figure 165: Bit ordering sequence for 64B/66B encoding 170
Figure 166: Scrambling 170
Figure 167: An example of 10GBASE-R implementation 171
Figure 168: Ethernet frame format 172
Figure 169: Format of Ethernet medium access control (MAC) address 174
Figure 170: Ethernet flow control (PAUSE frames) 175
Figure 171: An example of GbE MAC controller implementation 176
Figure 172: Role of the subnetwork access protocol (SNAP) 179
Figure 173: Medium dependent interface (MDI) 181
Figure 174: Attachment unit interface (AUI) 181
Figure 175: Medium independent interface (MII) 183
Figure 176: Reconciliation sublayer (RS) 184
Figure 177: Gigabit Ethernet (IEEE 802.3) 185
Figure 178: Re-using FIbre Channel specifications in Gigabit Ethernet 185
Figure 179: Gigabit medium independent interface (GMII) 186
Figure 180: Gigabit Interface Converter (GBIC) 187
Figure 181: 10 Gigabit Ethernet 188
Figure 182: 10 Gigabit Ethernet options 189
Figure 183: 10 Gigabit Ethernet sublayers 189
Figure 184: 10 Gigabit medium independent interface (XGMII) extension 190

Figure 185: Mapping of XGMII signals into XAUI 191
Figure 186: 10 Gigabit Ethernet physical layer (PHY) architecture 192
Figure 187: Gigabit Ethernet PCS reference diagram 193
Figure 188: 8B/10B encoder schematic 194
Figure 189: Schematic of an 8B/10B encoder/decoder implementation 195
Figure 190: Implementing the RESET_SYNC circuit (synchronous reset) . 196
Figure 191: Block diagram of the transmitter 197
Figure 192: Encoder block 198
Figure 193: Pipelining of the encoder 199
Figure 194: Block diagram of the receiver 200
Figure 195: State diagram of the SYNC_FSM block 201
Figure 196: Asynchronous transfer mode (ATM) basics 202
Figure 197: User-network interface (UNI) cell format 204
Figure 198: Network-network interface (NNI) cell format 205
Figure 199: UTOPIA interface 206
Figure 200: UTOPIA extended reference model 206
Figure 201: UTOPIA master and slave devices 207
Figure 202: UTOPIA level 1 (L1) specification 208
Figure 203: UTOPIA L1 cell format 210
Figure 204: Octet-level handshake 212
Figure 205: Cell-level handshake 212
Figure 206: Multiple PHY devices and ports 214
Figure 207: Multiplexed status polling (example 1) 215
Figure 208: Multiplexed status polling (example 2) 216
Figure 209: UTOPIA level 3 (L3) 216
Figure 210: Cell formats for 32-bit operation mode in UTOPIA L3 217
Figure 211: UTOPIA level 4 (L4) specification and signal definition 219
Figure 212: Source synchronous timing in UTOPIA L4 222
Figure 213: Fibre Channel architecture 224
Figure 214: Point-to-point topology 225
Figure 215: Arbitrated loop topology 225
Figure 216: FC fabric 227
Figure 217: Port assignments in FC 229
Figure 218: Open fiber control system (OFC) 231
Figure 219: Fibre Channel frame format 233
Figure 220: Fibre Channel frame header 233
Figure 221: Fibre Channel framing hierarchy 234
Figure 222: Buffer-to-buffer flow control 235
Figure 223: End-to-end flow control 236

Figure 224: Service classes 237
Figure 225: Fibre Channel components 242

LIST OF TABLES

Table 1: SCSI trade association-endorsed terminology 25
Table 2: Serial SCSI supports different transport technologies 29
Table 3: RapidIO bandwidth summary 63
Table 4: TFI-5 signal definitions 121
Table 5: Common parameters of line interfaces for different speeds 125
Table 6: Some examples of Ethernet standards 149
Table 7:Maximum distance in 10 Gigabit Ethernet 153
Table 8: 4N/5B coding 154
Table 9: 8B/6T encoding (8 binary/6 ternary) 156
Table 10: 8B/10B coding table 160
Table 11: 8B/10B control words 161
Table 12: 64B/66B code summary 163
Table 13: Control code mapping in 64B/66B code 163
Table 14: Link code word (LCW) 165
Table 15: Next page function 166
Table 16: LLC control field 177
Table 17: LLC address format (IEEE 802.2) 178
Table 18: LLC address 178
Table 19: Format of the SNAP header 179
Table 20: 10 Gigabit Ethernet interface summary 192
Table 21: Encoding of data by using two ROM lookup tables 199
Table 22: UTOPIA level differences 208
Table 23: PHY port address allocation 215
Table 24: Format of control information (Rx/Tx_Ctrl = 1) 222
Table 25: FC layers 223
Table 26: Arbitrated loop initialization 227
Table 27: Media types and speed options 230

LIST OF ABBREVIATIONS

AAL: ATM Adaptation Layer
AAL5: ATM Adaptation Layer 5
AC: Alternating Current
ACK: Acknowledgment
ADM: Add/Drop Multiplexer
ADR: Address
ADSL: Asymmetric Digital Subscriber Line
AIS: Alarm Indication Signal
ANSI: American National Standards Institute
ARB: Adaptive Rate Based
ASIC: Application Specific Integrated Circuit
ATA: Advanced Technology Attachment
ATM: Asynchronous Transfer Mode
AU: Administrative Unit (in SDH)
AUG: Administrative Unit Group (in SDH)
BER: Bit Error Rate
BIP-8: Bit Interleaved Parity of 8 bits
BIP-24: Bit Interleaved Parity of 24 bits
BNC: Bayonet Norm Connector
BSY: Busy
C/R: Command/Response indicator
CA: Collision Avoidance
CCITT: Comité Consultatif International Télégraphique et Téléphonique
CDR: Clock and Data Recovery
CEI: Common Electrical Interface
CLB: Configurable Logic Block
CLK: Clock
CMOS: Complementary Metal Oxide Semiconductor

CPU: Central Processing Unit
CRC: Cyclic Redundancy Check
CSIX: Common Switch Interface
CSMA/CD: Carrier Sense Multiple Access with Collision Detection
DA: Destination Address
DAS: Direct Attached Storage
DC: Direct Current
DDR: Double Data Rate
DMA: Direct Memory Access
DSAP: Destination Service Access Point
DXC: Digital Cross Connect
E/O: Electrical/Optical
EOP: End of Packet
EUI: Extended Unique Identifier
FC: Fibre Channel
FCIP: Fibre Channel over IP
FCP: Fibre Channel Protocol
FCS: Frame Check Sequence
FDDI: Fiber Distributed Data Interface
FEC: Forward Error Correction
FEXT: Far-End Cross Talk
FF: Flip Flop
FIFO: First In First Out
FPGA: Field Programmable Gate Array
FSM: Finite State Machine
FTP: File Transfer Protocol
GBIC: Gigabit Interface Converter
GE: Gigabit Ethernet
GFC: Generic Flow Control
GFP: Generic Framing Procedure
GHz: Gigahertz
GID: Global Identification
GMII: Gigabit Media Independent Interface
GSM: Global System for Mobile Communications
GbE: Gigabit Ethernet
HBA: Host Bus Adapter
HCA: Host Channel Adapter
HCS: Header Check Sequence
HDLC: High-Level Data Link Control