19

Title: HIPPI-6400 -- Designing for speed

CONF-980544--

Author(s): Donald E. Tolmie, CIC-5

Submitted to: 12th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'98), May 20-22, 1998, Edmonton, Alberta, Canada. If accepted, it will be part of a book published by Kluwer Academic Publishers.

HIPPI-6400 – Designing for speed

Don E. Tolmie
Los Alamos National Laboratory

ABSTRACT

The emerging High-Performance Parallel Interface – 6400 Mbit/s interface (HIPPI-6400), is targeted as a local area network (LAN), or system area network (SAN), supporting data rates of 6400 Mbit/s (800 Mbyte/s). This is eight times the speed of Gigabit Ethernet. The features used, and the design choices made, for the data link and physical layers of HIPPI-6400, to achieve this unprecedented speed are the subject of this paper. HIPPI-6400 borrowed freely from other successful technologies such as ATM, Ethernet and the original HIPPI – taking the best features of each and melding them with some new features. HIPPI-6400 is a cost effective reliable interconnect for distances up to 1 km; it intermixes large and small messages efficiently.

Keywords

HIPPI, gigabit, gigabyte, parallel, LAN, deskew

Contact person

Don E. Tolmie
Los Alamos National Laboratory
MS-B25, Los Alamos, NM 87545, USA
E-mail: det@lanl.gov
Phone: (505) 667-5502
FAX: (505) 665-7793

Background

The increasing complexity of server and cluster computing and bandwidth-hungry applications such as scientific computing, imaging, engineering modeling processing are demanding unprecedented interconnect speeds. Out of all the available gigabit and gigabyte technologies, Gigabit Ethernet, based on the framing Ethernet, has become the leading choice in meeting demands at a gigabit by offering greater bandwidth and improved client/server response times. Now, however, the emerging use of gigabit connections at the departmental server and desktop is creating a need for even higher-speed network technology at the backbone and in the cluster. The High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH), and the High-Performance Parallel Interface – 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC), are an answer to this need.[1,2] They will initially be deployed in a gigabyte system area network interconnecting high performance shared-memory multiprocessors (SMPs), clustered to provide an aggregate computing power – even beyond that achievable with the highest speed SMPs of today or tomorrow.

HIPPI-6400 represents the next generation beyond the current gigabit, and near gigabit, interconnect standards. Operating at 6400 Mbit/s, full-duplex, HIPPI-6400 ensures maximum compatibility with the Ethernet, Gigabit Ethernet, ATM, and HIPPI installed base. The original HIPPI standards, running at 800 and 1600 Mbit/s, developed and first deployed almost 10 years ago, pioneered higher speed interconnect technology. Along with a proposal from Silicon Graphics Inc., the original HIPPI provided the starting point for HIPPI-6400. HIPPI-6400 is based on the best features of several successful interfaces – drawing from ATM, Ethernet and the original HIPPI specifications. From ATM it borrowed a small 32-byte micropacket (like a 48-byte ATM cell), and four Virtual Circuits (fewer than ATM, but limited for performance reasons). From Ethernet it borrowed the MAC header to allow easy translation to other popular protocols, and to use existing Ethernet-based control and management tools. From the original HIPPI it borrowed the large message size capability, credit-based flow control, encoding scheme for dc-balance, and a cable using multiple twisted-pairs (or optical fibers), for the data path. Features of HIPPI-6400 not found in any of these interfaces include end-to-end as well as link-level checksums, automatic retransmission at the physical layer to correct flawed data, and a data rate of 6400 Mbit/s. As in other gigabit technologies, HIPPI-6400 systems will be switched rather than have multiple devices sharing a common bus or medium.

The HIPPI-6400 standards are being developed in ANSI Task Group T11.1 (see the web page at **http://www.cic-5.lanl.gov/~det** for meeting notices, meeting minutes, and draft documents). In relation to the OSI Reference Model, HIPPI-6400-PH (Physical Layer) specifies the physical and data link layers. HIPPI-6400-SC (Physical Switch Control) specifies a network layer for controlling physical layer switches. T11.1 completed their work on these documents in October 1997, and forwarded them for further review and balloting. The HIPPI-6400-PH and -SC documents are expected to complete their processing and become approved ANSI standards in late 1998. In addition, Task Group T11.1 is working on a transport layer standard, initially part of HIPPI-6400-PH, called the Scheduled Transfer Protocol (ST). Scheduled Transfer takes advantage of the high-speed reliable HIPPI-6400 lower layers, and provides additional performance by bypassing parts of the host's operating system. Scheduled Transfer specifies mappings for use on Ethernet, ATM, and Fibre Channel, as well as HIPPI-6400.

System features

Figure 1 shows a system overview with a HIPPI-6400 switch interconnecting four nodes, two of which are translators to other media (e.g, to Gigabit Ethernet to talk to Ethernet-based devices in a local environment, and to ATM to connect to other far-flung sites over the telephone network). The networking aspects of HIPPI-6400 are detailed in the HIPPI-6400-SC document.

HIPPI-6400-PH defines a symmetric point-to-point physical link for transferring micropackets. The physical links are bi-directional and capable of the full 6400 Mbit/s bandwidth in both direction simultaneously. The logical links are simplex, i.e., the data inbound and outbound are completely separate. A link's control information is carried on separate wires in parallel with the user's data (i.e., out-of-band). The control information is not counted in the 6400 Mbit/s bandwidth number (i.e., the rate available for the user's data is 99.6% of the 6400 Mbit/s).

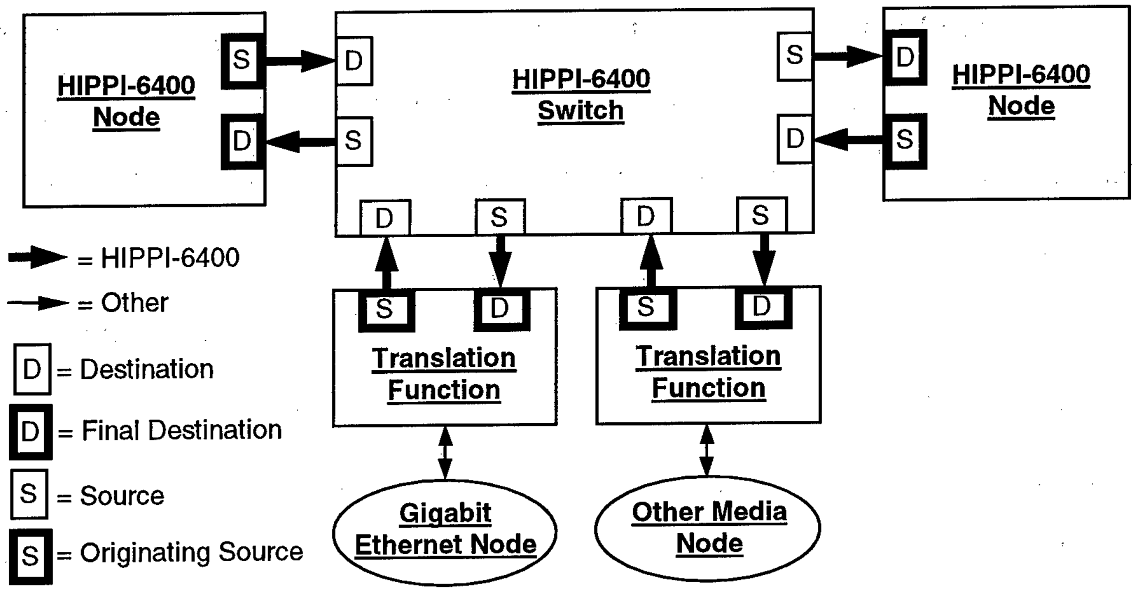

Figure 1 – System overview

Virtual Channels

Four Virtual Channels, VC0, VC1, VC2, and VC3, are available in each direction on each link. The VCs are assigned to specific message sizes and transfer methods. All of the micropackets of a message are transmitted on a single VC, i.e., the VC number does not change as the micropackets travel from the Originating Source to the Final Destination over one or more links. Messages to a Final Destination are delivered in order on a single VC.

Worm-hole routing is used in the HIPPI-6400 switches rather than the virtual connections used in ATM, or the end-to-end connections used in the original HIPPI. Worm-hole routing means that a message is sent into the network without prior knowledge if a free path is currently available to the Final Destination. If the message hits a link (e.g., on the output of a switch), that is using the same Virtual Channel, then the new message must wait for the existing message to complete (Tail bit = 1), before progressing further. On the plus side, worm-hole routing does not need time-consuming circuit setup or teardown, or for the links and switches to maintain large amounts of state information.

The VCs provide a multiplexing mechanism which can be used to prevent a large message from blocking a small message until the large message has completed; in contrast to the original HIPPI where a large message blocked any messages queued behind it. The number of Virtual Channels was deliberately limited to four (as opposed to the almost unlimited number in ATM), since for performance reasons the buffering needed to be on-chip. Three message sizes are supported; VC0 $\leq$ 2176 bytes, VC1 $\leq$ 128 Kbytes, VC2 $\leq$ 128 Kbytes, and VC3 $\leq$ 4 Gbytes. The intent was to separate the small control messages from the larger messages, i.e., as shown by the bi-modal packet sizes in most networks.

Micropackets

Micropackets are the basic transfer unit from Source to Destination on a link. As shown in figure 2, a micropacket is composed of 32 data bytes and 8 bytes of control information. This small transfer unit (i.e., the micropacket), results in a low latency for short messages and a component for large transfers. At 6400 Mbit/s, a micropacket is transmitted every 40 ns, with Null micropackets transmitted when other micropackets are not available. Credit and retransmit operations are performed on a micropacket basis.

	Control Information (8 bytes)	
	bits	**Function**
User data (32 bytes)	4	Micropacket Type
	2	Virtual Channel selector
	8	Transmit sequence number
	8	Receive sequence number (i.e., ACK)
	1	Tail bit (i.e., End of Message)
	1	Error (i.e., upstream unrecoverable error)
	6	Credit update value
	2	Virtual Channel number for credit update
	16	End-to-end CRC $(x^{16} + x^{12} + x^5 + 1)$
	16	Link lever CRC $(x^{16} + x^{12} + x^3 + x + 1)$

Figure 2 – Micropacket contents

Micropacket Type	Data byte contents	Micropacket carries : Transmit sequence #	Receive sequence #	Credit update
Header	24-byte Header and 8 bytes of user data	Yes	Yes	Yes
Data	32 bytes of user data	Yes	Yes	Yes
Admin	Admin message	Yes	Yes	Yes
Credit-only	0's	Yes	Yes	Yes
Null	0's	Invalid	Yes	Invalid
Reset or Initialize	0's	Invalid	Invalid	Invalid

Figure 3 – Capabilities of each Type of micropacket

Messages

A message is an ordered sequence of one or more micropackets which have the same VC, Originating Source, and Final Destination. Messages carry the payload data. The first micropacket of a message, i.e., the Header micropacket, contains a HIPPI-6400 Header (i.e., 24 bytes of information used to route through a HIPPI-6400 fabric), and 8 bytes of user data. The last micropacket of the message is marked with the Tail bit (much like an ATM AAL5 packet).

The contents of a HIPPI-6400 Header are shown in figure 4. The MAC header is the same as the IEEE 802.3 header except that the length field (M_len) is 32 bits in HIPPI-6400 for longer messages, while in IEEE 802.3 it is 16 bits. The D_ULA and S_ULA are the 48-bit IEEE Universal LAN Addresses for the Originating Source and Final Destination. The IEEE 802.2 LLC/SNAP header is used to carry the EtherType, which selects the upper-layer protocol. Translating to other common networks, e.g., Gigabit Ethernet, is facilitated by using the common IEEE network formats .

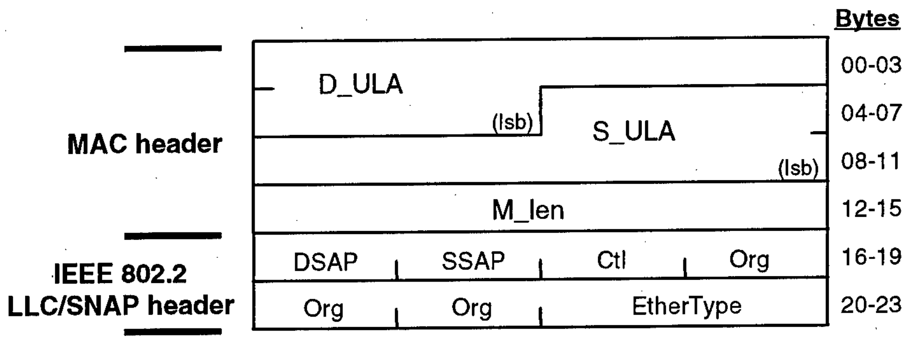

Figure 4 – HIPPI-6400 Header

Figure 5 shows a Message contained in five micropackets. Bytes $N - M$ are the user payload bytes. If a Message does not end on a micropacket boundary, then pad bytes of zeros are included in the last micropacket.

Micropacket number	Data Bytes contents	Tail bit
1	Header, Bytes 0 - 7	0
2	Bytes 8 - 39	0
3	Bytes 40 - 71	0
4	Bytes 72 - 103	0
5	Bytes 104 - 135	1

Figure 5 – Message contained in five micropackets

Flow Control

Link-level credit-based flow control is used between a Source and Destination to prevent overrunning a Destination's buffers. Note that the flow control is between a Source and Destination, not necessarily the Originating Source and Final Destination (see figure 1). As shown in figure 6, the credits are assigned on a VC basis, i.e., VC0's credits are separate from VC1's credits (hence congestion on VC3 will not stall traffic on VC0). The Destination end of a link grants credits to match the number of free receive buffers for a particular VC. The Source end of the link consumes credits as it moves micropackets from the VC Buffers to the Output Buffer. Note that flow control is on a link basis, i.e., hop-by-hop. If a link has credit information, but no data, to transmit, then "credit-only" micropackets are transmitted. The micropackets containing credit information are checked for delivery, and included in the retransmission if an error occurs. Credit information in the original HIPPI was not as reliable, and in error cases could be lost – possibly leading to credit starvation; this is not possible in HIPPI-6400-PH. We feel that credit-based flow is the optimum method in a local area network environment where the distances are short and the buffering limited, but in a wide area network environment rate-based control is preferred.

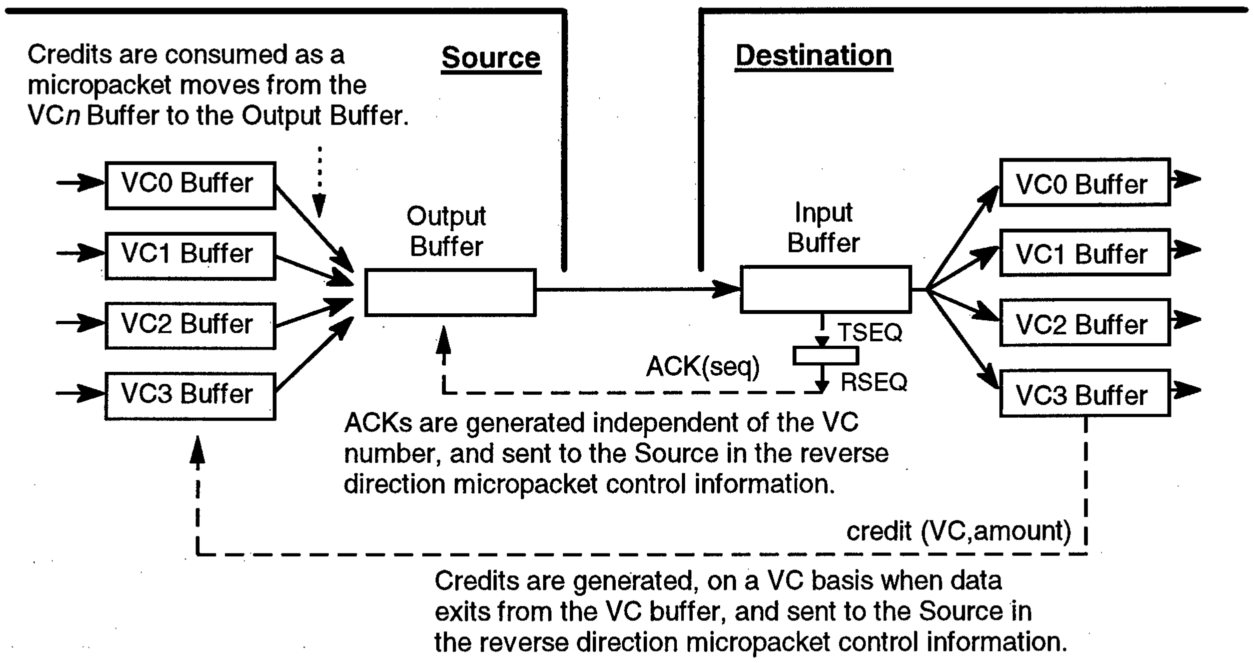
Credits are consumed as a micropacket moves from the VC*n* Buffer to the Output Buffer.
Source
Destination
VC0 Buffer
VC1 Buffer
VC2 Buffer
VC3 Buffer
Output Buffer
Input Buffer
TSEQ
ACK(seq)
RSEQ
VC0 Buffer
VC1 Buffer
VC2 Buffer
VC3 Buffer
ACKs are generated independent of the VC number, and sent to the Source in the reverse direction micropacket control information.
credit (VC,amount)
Credits are generated, on a VC basis when data exits from the VC buffer, and sent to the Source in the reverse direction micropacket control information.

Figure 6 – Reverse direction control information

It was the permissible buffer size that limited the link to 1 km without speed degradation. For performance reasons the Destination buffers had to be on-chip, and about 10 KBytes was available for each of the four VCs. At 6400 Mbit/s (800 Mbyte/s), 5 ns/m propagation delay, and 10 KBytes in flight (assuming the worst case with all of the in-flight data directed to a single receive buffer), the distance can be calculated as 2.5 km. The 2.5 km is a round trip distance (giving time for acknowledgements to get back to the Source), and does not include any processing overhead. Hence, the link distance was specified as 1 km maximum; the speed may decrease at greater distances. Note that the distance limit, before speed degradation, is dependant on fully loading a single VC with data; spreading the load over multiple VCs, or not trying to send at full rate, gives longer distances.

Retransmission

Retransmission is performed to correct flawed micropackets; providing in-order, reliable data delivery. Go-back-N retransmission is used, i.e., if an error is detected then the flawed micropacket, and all micropackets transmitted after it, are retransmitted. The CRCs in each micropacket are checked at the Destination side of a link; at the Input Buffer in figure 6. Correct micropackets are acknowledged, flawed micropackets are discarded. Note that retransmission is independent of the VC used, and also independent of the credit information, i.e., retransmission occurs between the Output and Input Buffers in figure 6 while VC and credit information pertains only to the VC Buffers. Retransmission is on a link basis, i.e., hop-by-hop.

Sequence numbers, in a micropacket's control information, are transmitted with all micropackets that contain data or credit information (other micropackets, e.g., Type = Null, use sequence number = x'FF'). The receiver acknowledges micropackets by returning the highest sequence number of contiguously good micropackets. Hence, if a micropacket is received in error, the receive sequence number sticks on the value of the last correct micropacket. A timeout mechanism at the sender detects that a transmitted micropacket was not acknowledged, and retransmits all micropackets starting with the one in error. Note that only micropackets with transmit sequence numbers (see figure 3) are retransmitted. The timeout mechanism was chosen because it was more robust than sending an ACK, i.e., if an ACK is dropped the protocol will just wait for the next ACK. A timeout mechanism may not be appropriate for a link with a long delay, but is preferred when the link delay is low (on the order of 10 μs for HIPPI-6400), and adequate buffering is available. The 8-bit sequence numbers allow up to 256 unacknowledged micropackets, i.e., 10 KBytes or the size of the receive buffer.

Check functions

Two 16-bit cyclic redundancy checks (CRCs), with different polynomials, are used. The LCRC is the link-level checksum; the ECRC is the end-to-end checksum. Figure 7 shows a 5-micropacket Message, and the coverage for each CRC. Bytes *N* – *M* are the user payload, c00–c47 are the first 48 control bits, and c48–c63 contain the ECRC and LCRC (see figure 2).

Micropacket number	Data Bytes contents	LCRC checksum coverage	ECRC checksum coverage
1	Header, Bytes 0–7	Header, Bytes 0–7, c00–c47	Header, Bytes 0–7
2	Bytes 8–39	Bytes 8–39, c00–c47	Header, Bytes 0–39
3	Bytes 40–71	Bytes 40–71, c00–c47	Header, Bytes 0–71
4	Bytes 72–103	Bytes 72–103, c00–c47	Header, Bytes 0–103
5	Bytes 104–135	Bytes 104–135, c00–c47	Header, Bytes 0–135

Figure 7 – Checksum coverage for a 5-micropacket Message

The end-to-end CRC (ECRC) covers the data bytes of all of the micropackets in a Message, i.e., the Header micropacket and all of the Data micropackets (if any) up to this point in a Message. The ECRC does not cover the control bits. The ECRC is unchanged from the Originating Source to the Final Destination, e.g., through switches and bridges. The ECRC is accumulated over an entire Message, i.e., it is not re-initialized for intermediate Data micropackets. Note that in figure 7, the second micropacket's ECRC covers the information in the first and second micropacket; the third micropacket's ECRC covers the information in the first, second, and third micropacket, etc. The ECRC generator polynomial is:

$$x^{16} + x^{12} + x^3 + x + 1$$

The link CRC (LCRC) covers all of the data and control bits of a micropacket, with the exception of itself. The LCRC is initialized for each micropacket, and must be calculated fresh for each link since some values change hop-to-hop, e.g, Received sequence number and credit information. The LCRC polynomial is:

$$x^{16} + x^{12} + x^5 + 1$$

Both CRCs are checked at each HIPPI-6400 node, be it a switch or end device. The combination of two 16-bit CRCs provides a stronger check than a single 16-bit CRC for link-level checking of individual micropackets. Analysis has shown that there are no undetected errors unless at least 6 bits are in a micropacket are in error (i.e., 1 in 1.86 billion bits).[3] Not only must there be at least 6 bits in error, but the bits must be strategically located and not contiguous.

In addition, the two separate CRCs are easier to calculate than a single 32-bit CRC. While many CRC implementations are done in a serial bit-by-bit fashion, at the speeds of HIPPI-6400 this may not be feasible. As an aid to the designer, example circuits and equations for parallel CRC implementations are included in an informative annex in HIPPI-6400-PH.[1]

The Error bit in Data micropackets is used to inform downstream HIPPI-6400 nodes that an uncorrectable error occurred upstream, for example from a translator to another media that does not provide retransmissions. Received Data micropackets with the Error bit set are passed on and not reported. This helps pinpoint where the error occurred; it would be next to impossible if everyone downstream also reported the error.

A Source also has the capability to abort a micropacket by forcing a specific LCRC value (called a "stomp code"). Downstream HIPPI-6400 nodes receiving a stomped micropacket will discard it as if were a Null micropacket. Other checks are made for out-of-order or missing micropackets (e.g., two Header micropackets without an intermediate Tail bit), lack of credit for a timeout period, etc. All error events are logged. There are no know error cases that would cause a link to lock up. An upper-layer protocol only needs to retransmit those messages that had unrecoverable errors, and these should be few and far between on a properly installed and maintained HIPPI-6400 system.

Media interfaces

The data is transmitted in parallel over the cable, and strobed with the clock signal. Figure 8 shows the signal lines between two end devices. Figure 9 shows the signal waveforms during a micropacket time (all of the time except the 40 ns when retraining the deskew circuitry).

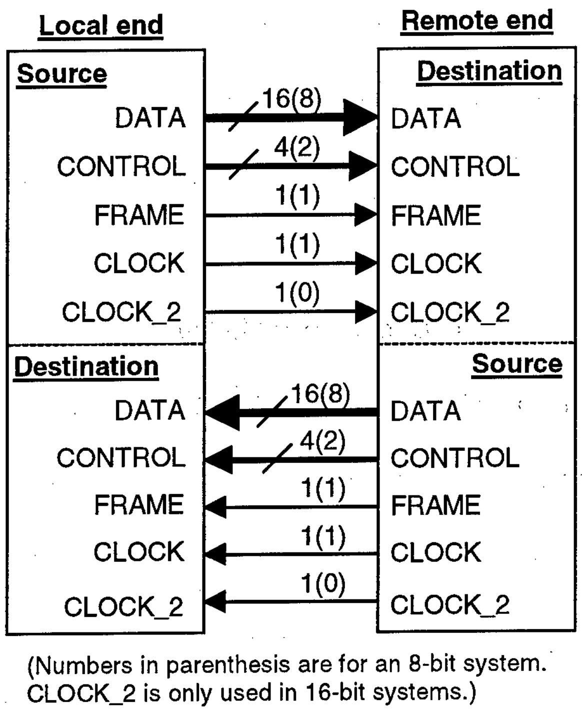

Figure 8 – HIPPI-6400-PH link showing signal lines

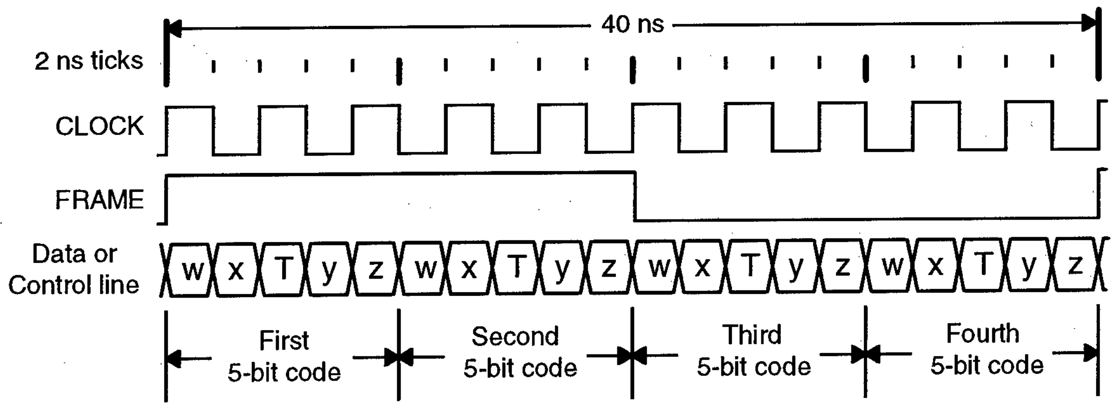

Figure 9 – 16-bit system micropacket waveforms

The parallel architecture allowed the use of CMOS circuits and available drivers and receivers, a real cost and time-to-market saving. A serial implementation of HIPPI-6400 would have required a serial rate of about 10 Gbit/s, costly with optics, and impossible with copper cable.

A copper cable interface is defined for the 16-bit system, using a total of 23 signals in each direction. Each signal operates at 500 MBaud. The cable assembly (i.e., cable and connectors) provides differential paths for 46 signals, 23 in each direction. Characteristic impedance is 150 Ω and the maximum distance supported is 40 m. The cable to support this speed and distance is not cheap, but is available from several vendors. Some testing has shown that passive equalizers aid signal quality for cables greater than 10 m. Active equalizers would have given longer distances, but required power, took considerable room, and added cost.

A local electrical interface is also defined, with the intent to drive parallel optical transceivers on the same circuit board. The optical interface is defined for an 8-bit system, with a total of 12 signals in each direction (see figure 8). Each signal operates at 1 GBaud. A 12-fiber ribbon cable is used in each direction. The optical interface is not as far along in design and standardization, and has been split out into a separate standards document called High-Performance Parallel Interface – 6400 Mbit/s Optical Specification (HIPPI-6400-OPT). [4] Several optical variants are being explored. One uses 850 nm laser arrays, 62.5/125 μm multimode fiber, and an open-fiber-control system to detect an open fiber and power down the lasers (to avoid potential eye damage). Another variant uses the same lasers and fiber, but decreases the power to avoid eye safety problems. The third variant uses 1300 nm lasers and either single-mode or multimode fiber. The human eye is much less susceptible to the 1300 nm wavelength, and that system will probably not need an open fiber control safety system. The 850 nm variants will probably be limited to 200–300 m, while the 1300 nm variant with single-mode fiber may operate up to 10 km. The HIPPI-6400-OPT specification is being written with the intent that it can also be used for other systems needing high-speed parallel fiber paths.

AC coupling

When driving long cables it is usually desirable to AC couple the signals and to keep them DC balanced. The AC coupling separates the ground paths between the end devices and avoids ground loops. DC balance means that a signal is above the switching threshold as much of the time as it is below the threshold; this considerably improves jitter and signal quality. 4B/5B encoders / decoders are specified in HIPPI-6400-PH, one encoder / decoder on each signal line. The 4B/5B encoding is adapted from the HIPPI-Serial standard. [4, 5, 6] The 4B/5B encoding scheme transmits four data bits as a 5-bit code group. The 4B/5B encoding was chosen for its implementation simplicity since 20 copies are required on the chip. Figure 10 is a simplified schematic of an encoder on the left, and a decoder on the right.

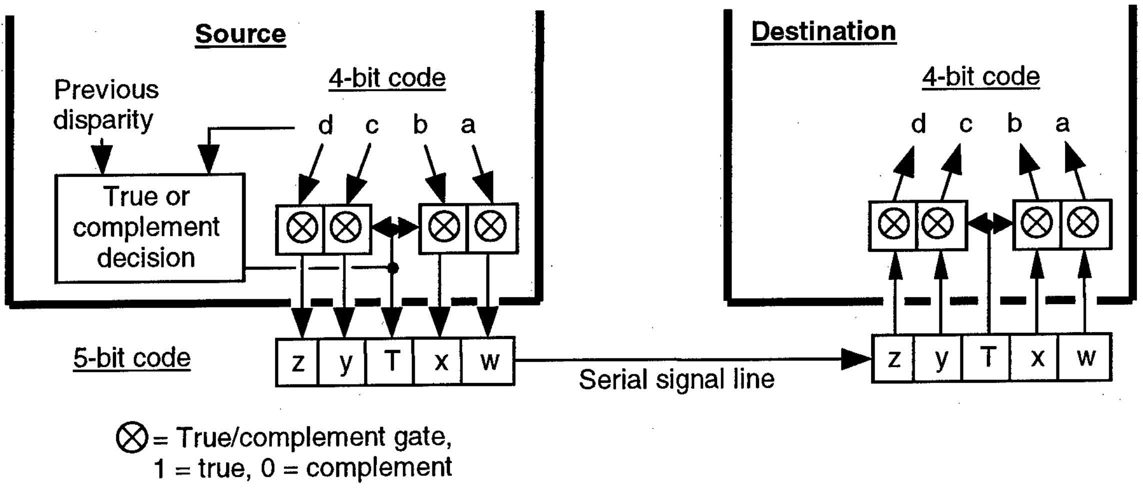

Figure 10 – 4B/5B Encoder / Decoder

For each signal line, a running count, called the Disparity Count, is kept of all the ones and zeros transmitted on that line since the link was reset. The Disparity Count is incremented for each "1" transmitted, and decremented for each "0" transmitted. The 5-bit code (w,x,T,y, and z in figure 10) transmitted is based on the current value of the Disparity Count and the input data 4-bit code (a, b, c, and d in figure 10). For example, if the Disparity Count is negative (more 0's than 1's transmitted), and the incoming 4-bit data has more 0's than 1's, then the incoming 4-bit code is complemented (generating more 1's), and the "T" bit set to 0. At the receive end the incoming bits are passed straight through (if T = 1), or complemented (if T = 0). This algorithm gives a maximum run length of 11 bits, and a maximum disparity of +6 and -7. While the run length and maximum disparity are not as good as the 8B/10B code used in Fibre Channel [7,8], the 4B/5B algorithm is much simpler to implement, and simplicity is mandatory when you remember that a single link requires 20 copies of the circuit (one for each data and control bit line).

A design goal for the 4B/B encoding was to minimize the average run length for real data. As a test case, the operating system of a Silicon Graphics workstation was used as the random data input for a 4B/5B simulator. Rather than in the middle of a 5-bit data pattern, the "T" bit had started out on the end. It turned out that the operating system had many 4-bit zero patterns (i.e., binary 0000), and these would give long run lengths when the zeros were back-to-back. Moving the "T" bit to the center of the 5-bit code shortened the average run length considerably, moving the worst case 4-bit code combination from "0000" to "1111". Since real user data is more likely to contain 0000 rather than 1111 patterns, this move was considered useful for the general case.

Deskewing the parallel signals

The clock signal, used to strobe the other signals, is carried on a separate line, negating the need for clock recovery circuits on each data line. Up to 10 ns of differential skew is allowed between the signals lines at the receiver, and the deskew circuits are dynamically adjusted every 10 microseconds. The deskew adjustment eats up one micropacket time (i.e., 40 ns), every 10 μs, accounting for the missing 0.04% of the 6400 Mbit/s total bandwidth. Figure 11 is a block diagram of the deskew circuit on one signal line, there are a total of 20 such circuits on an interface chip. The input signal drives a tapped delay line (implemented as a series of inverters), and the output is derived from one of the taps. A special signal pattern is used to train the deskew logic.

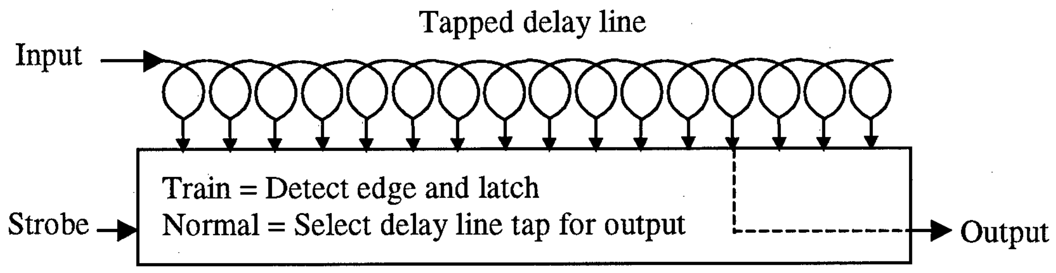
Tapped delay line
Input
Strobe
Train = Detect edge and latch
Normal = Select delay line tap for output
Output

Figure 11 – Tapped delay line deskew circuit

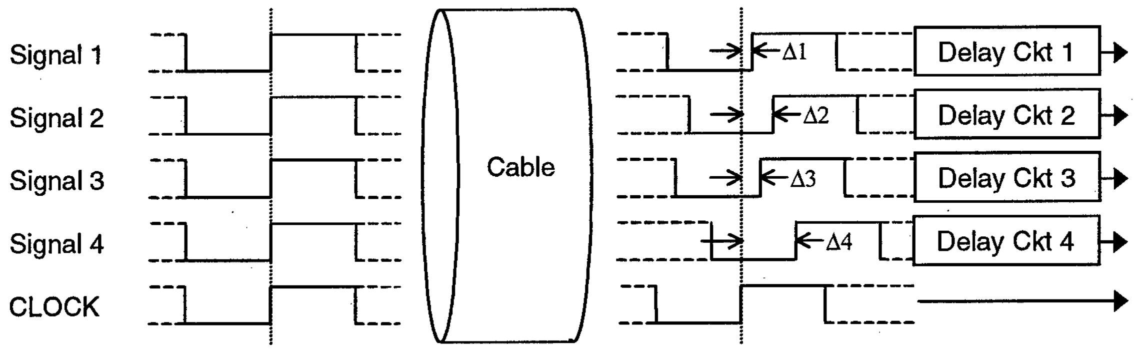

Figure 12 – Dynamic deskew in operation

Figure 12 shows four signals being deskewed. They are transmitted in synchronism on the left, but coming out of the cable they are skewed due to differences in wire lengths, propagation delay, etc. Delay Ckt 1 is adjusted by Δ1, Delay Ckt 2 by Δ2, etc., so that all of the signals are again synchronous as they leave the delay circuits providing the deskew function.

Summary

HIPPI-6400 is an emerging standard for moving digital data at speeds of up to 6400 Mbit/s (800 Mbyte/s) between devices in a LAN-like environment. Many innovative design techniques are employed, resulting in a robust full-duplex link with efficient, reliable, in-order, data delivery. The links use parallel copper or fiber paths so that today's CMOS technology can be used to implement the links.

Acknowledgements

The Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36. The author's work was performed under auspices of the U.S. Department of Energy. Silicon Graphics Inc., with Dr. Greg Chesson leading their efforts, has contributed the majority of the HIPPI-6400 technical innovations. The HIPPI standards committee, with participation from many people throughout the industry, has worked tirelessly to document HIPPI-6400 and Scheduled Transfer as ANSI standards.

References

[1] ANSI X3.xxx-199x, High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH).

[2] ANSI X3.xxx-199x, High-Performance Parallel Interface – 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC).

[3] HIPPI-6400: Analysis of a High-Throughput Network Interface, James Hoffman, Master's Thesis, University of Arizona, 1996.

[4] ANSI X3.300-1997, High-Performance Parallel Interface – Serial Specification (HIPPI-Serial).

[5] "DC-Free Code for Arbitrary Data Transmission", Doug Crandall, Steve Hessel, Tom Hornak, Rasmus Nordby, Kent Springer and Richard C. Walker, U.S. Patent 5438621 (August 1995).

[6] "DC-Free Line Code and Bit and Frame Synchronization for Arbitrary Data Transmission", Tom Hornak, Benny Lai, Pat Petruno, Cheryl Stout, Rick Walker, Jieh-Tsorng Wu, Chu Yen, U.S. Patent 5022051 (June 1991).

[7] "A DC-Balanced, Partioned-Block, 8B/10B Transmission Code", A.X. Widmer and P.A. Franaszek, *IBM Journal of Research and Development,* 27, No.5: 440-451 (September, 1983).

[8] "Byte-Oriented DC Balanced (0,4) 8B/10B Partitioned Block Transmission Code", P. A. Franaszek and A.W. Widmer, U.S. Patent 4486739 (September 1983).

Biography

Don Tolmie joined the Los Alamos National Laboratory in 1959, and has been involved with networking of supercomputers since 1972. He has led the HIPPI standards efforts since HIPPI's conception in 1987, and is presently the Chairman of T11.1. He holds a BSEE from New Mexico State University (1959), and an MSEE from University of California - Berkeley (1961).

M98004383

Report Number (14)	LA-UR--97-4906
	CONF-980544--
Publ. Date (11)	199803
Sponsor Code (18)	DOE/DP, XF
UC Category (19)	UC-705, DOE/ER

DOE