16 HIPPI-6400—DESIGNING FOR SPEED

Don E. Tolmie

Los Alamos National Laboratory,
Los Alamos, U.S.A.
det@lanl.gov

Abstract: The emerging High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH) is targeted as a local area network (LAN) or system area network (SAN), supporting data rates of 6400 Mbit/s (800 Mbytes/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 kilometer; it intermixes large and small messages efficiently.
Keywords: HIPPI, gigabit, gigabyte, parallel, LAN, deskew.

16.1 BACKGROUND

The increasing complexity of server and cluster computing, and bandwidth-hungry applications such as scientific computing, imaging, and modeling, 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 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC), are an answer to this need (ANSI, 1998). 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, a data rate of 6400 Mbit/s, and very low latency. 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/lanp/ANSI/ 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.

16.2 SYSTEM FEATURES

Figure 16.1 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 bidirectional and capable of the full 6400 Mbit/s

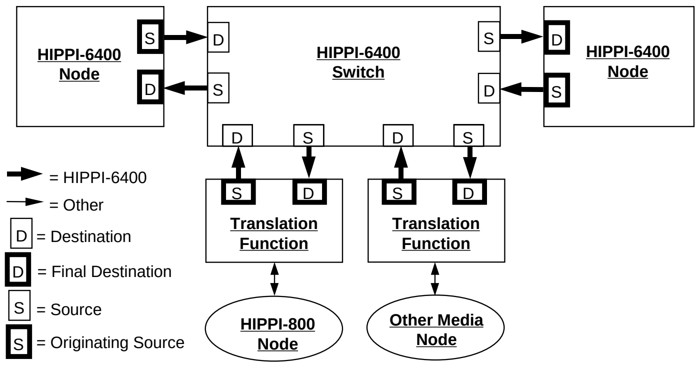

Figure 16.1 System overview.

bandwidth in both direction simultaneously. The logical links are simplex; 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 (the rate available for the user's data is 99.6% of the 6400 Mbit/s).

16.3 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; 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 the buffering needed to be on-chip for performance reasons. Three message sizes are supported: VC0 $\leq$ 2,176 bytes, VC1 $\leq$ 128 KB, VC2 $\leq$ 128 KB, and VC3 $\leq$ 4 GB. 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).

16.4 MICROPACKETS

Micropackets are the basic transfer unit from Source to Destination on a link. As shown in Table 16.1, a micropacket is composed of 32 data bytes and 8 bytes of control information. This small transfer unit (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 nanoseconds, with Null micropackets transmitted when other micropackets are not available. Credit and retransmit operations are performed on a micropacket basis.

Table 16.1 Micropacket contents.

	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 (ACK)
	1	Tail bit (end of message)
	1	Error (unrecoverable upstream 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 level CRC ($x^{16} + x^{12} + x^3 + x + 1$)

Table 16.2 details the different micropacket Types, and the Data byte contents for each Type. In addition, the control fields carrying flow control information are detailed as to whether the field carries a valid value for that micropacket Type. A field with an invalid value is ignored.

16.5 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, the Header micropacket, contains a HIPPI-6400 Header (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).

Table 16.2 Capabilities of each type of micropacket.

Micropacket Type	Micropacket carries:			
	Data byte contents	Transmit sequence #	Receive sequence #	Credit update
Header	24-byte Header and 8 bytes of user date	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

The contents of a HIPPI-6400 Header are shown in Figure 16.2. 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 Ether-Type, which selects the upper-layer protocol. Translating to other common networks is facilitated by using the IEEE network formats.

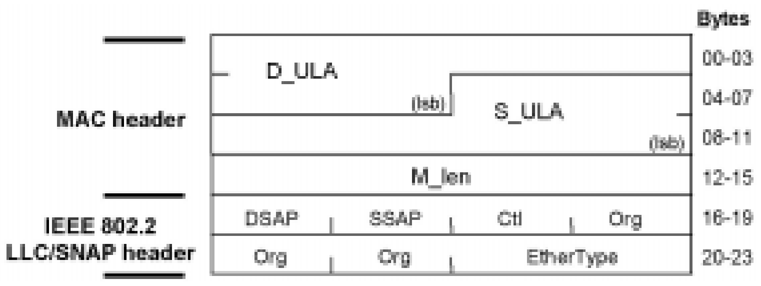

Figure 16.2 HIPPI-6400 header.

Table 16.3 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, the last micropacket is padded with zeroes.

16.6 FLOW CONTROL

Link-level credit-based flow control is used between a Source and Destination to prevent over-running 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 16.1).

Table 16.3 Message contained in five micropackets.

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

As shown in Figure 16.3, the credits are assigned on a VC basis; 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.

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.

It was the permissible buffer size that limited the link to one kilometer without speed degradation. For performance reasons the Destination buffers had to be on-chip, and about 10 KB was available for each of the four VCs. At 6400 Mbit/s (800 Mbytes/s), 5 nanoseconds per meter propagation delay, and 10 KB 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 kilometers.

The 2.5 kilometer is a round trip distance (giving time for acknowledgments to get back to the Source), and does not include any processing overhead. Hence, the link distance was specified as one kilometer maximum; the speed may decrease at greater distances. Note that the distance limit, before speed degradation, is dependent on fully loading a single VC with data. Spreading the load over multiple VCs or not trying to send at the full rate gives longer distances.

16.7 RETRANSMISSION

Retransmission is performed to correct flawed micropackets; providing in-order, reliable data delivery. Go-back-N retransmission is used; 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 16.3. Correct micropackets are acknowledged, flawed micropackets

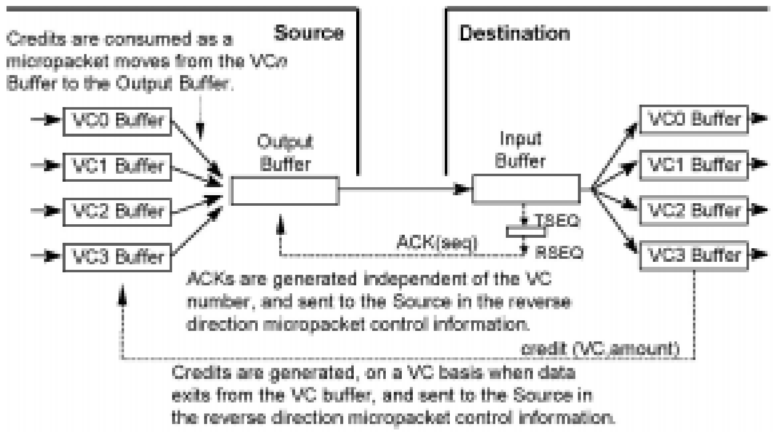
Source
Destination
Credits are consumed as a micropacket moves from the VCn Buffer to the Output Buffer.
VC0 Buffer
VC1 Buffer
VC2 Buffer
VC3 Buffer
Output Buffer
Input Buffer
TSEQ
RSEQ
ACK(seq)
ACKs are generated independent of the VC number, and sent to the Source in the reverse direction micropacket 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 16.3 Reverse direction control information.

are discarded. Note that retransmission is independent of the VC used, and also independent of the credit information. That is, retransmission occurs between the Output and Input Buffers in Figure 16.3, while VC and credit information pertains only to the VC Buffers. Retransmission is on a link basis.

Sequence numbers, in a micropacket's control information, are transmitted with all micropackets that contain data or credit information. Other micropackets, such as Type = Null, use hexadecimal sequence number ‘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 Table 16.2) are retransmitted. The timeout mechanism was chosen because it was more robust than sending an ACK; 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 microseconds for HIPPI-6400), and adequate buffering is available. The 8-bit sequence numbers allow up to 256 unacknowledged micropackets (10 KB, the size of the receive buffer).

16.8 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. Table 16.4 shows a 5-micropacket message, and the coverage for each CRC. Bytes x–y are the user payload; c00-c47 are the first 48 control bits, and c48-c63 contain the ECRC and LCRC (see Table 16.1).

Table 16.4 Checksum coverage for a 5-micropacket message.

Packet 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

The end-to-end CRC (ECRC) covers the data bytes of all of the micropackets in a message, which includes 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; it is not re-initialized for intermediate Data micropackets. Note that in Table 16.4, 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 in a micropacket are in error (1 in 1.86 billion bits) (Hoffman, 1996). 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 (ANSI, 1998).

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 known 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.

16.9 LATENCY

The Silicon Graphics Inc. SuMAC chip, which implements HIPPI-6400-PH, has shown latencies of 90 nanoseconds in one direction, and 120 nanoseconds in the other direction, for a total end-to-end latency of 210 nanoseconds. A switch path between two hosts would most likely contain two SuMAC chips (one for input and one for output). In addition, a switch may service up to 69 micropackets on each of the other three Virtual Channels before getting to your Virtual Channel (giving a worst case total of 207 micropackets). At 40 nanoseconds per micropacket, and a cable delay of about 1.5 nanoseconds per meter, this translates to a worst case latency of about 10 microseconds. Typical latencies should be on the order of 1 microseconds.

16.10 MEDIA INTERFACES

The data is transmitted in parallel over the cable, and strobed with the clock signal. Figure 16.4 shows the signal lines between two end devices. Figure 16.5 shows the signal waveforms during a micropacket time (all of the time except the 40 nanoseconds when retraining the deskew circuitry).

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 (cable and connectors) provides differential paths for 46 signals, 23 in each direction. The cable's characteristic impedance is 150 Ω and the maximum distance supported is 40 meters. 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 meters. Active equalizers would have given longer distances, but they 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

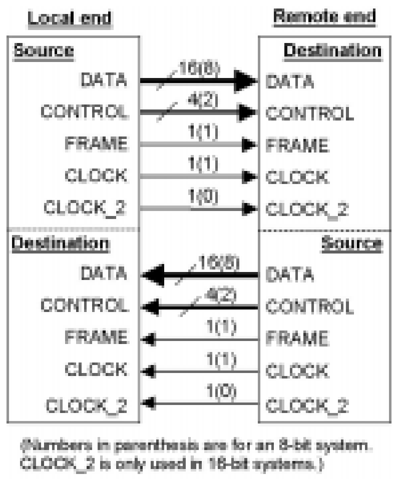
Local end
Remote end
Source
Destination
DATA
CONTROL
FRAME
CLOCK
CLOCK_2
16(8)
4(2)
1(1)
1(0)
(Numbers in parenthesis are for an 8-bit system.
CLOCK_2 is only used in 16-bit systems.)

Figure 16.4 HIPPI-6400-PH link showing signal lines.

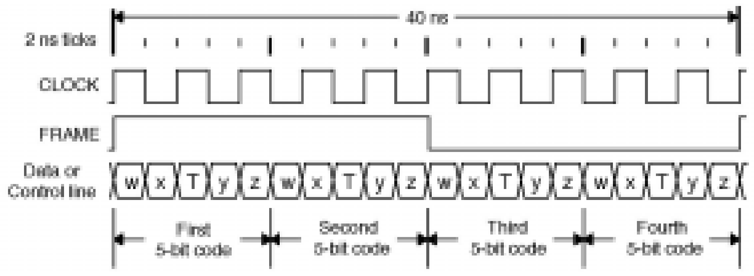

Figure 16.5 16-bit system micropacket waveforms.

system, with a total of 12 signals in each direction (see Figure 16.4). 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 the High-Performance Parallel Interface—6400 Mbit/s Optical Specification (HIPPI-6400-OPT) (ANSI, 1998). Several optical variants are being explored. One uses 850 nanometer laser arrays, 62.5/125 micron multimode fiber, and may use 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 nanometer lasers and either single-mode or multimode fiber. The human eye is much less susceptible to the 1300 nanometer wavelength, and that system will probably not need an open fiber control safety system. The 850 nanometer variants will probably be limited to 200–300 meters, while the 1300 nanometer variant with single-mode fiber may operate up to 10 kilometers. 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.

16.11 AC COUPLING

When driving long cables it is highly 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. HIPPI-6400-PH specifies the 4-bit to 5-bit (4B/5B) encoders/decoders, one encoder/decoder on each signal line. The 4B/5B encoding is adapted from the HIPPI-Serial standard (ANSI, 1997) and derived from some U.S. Patents (Crandall *et al.*, 1995; Hornak *et al.*, 1991). The 4B/5B encoding scheme transmits four data bits as a 5-bit code group (w, x, T, y, and z in Figure 16.5). The 4B/5B encoding was chosen for its implementation simplicity since 20 copies are required on the chip.

Figure 16.6 is a simplified schematic. A 4-bit to 5-bit encoder is shown on the left, and a 5-bit to 4-bit decoder is on the right. 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 transmitted (w, x, T, y, and z in Figure 16.6), is based on the current value of the Disparity Count and the input data 4-bit code (a, b, c, and d in Figure 16.6).

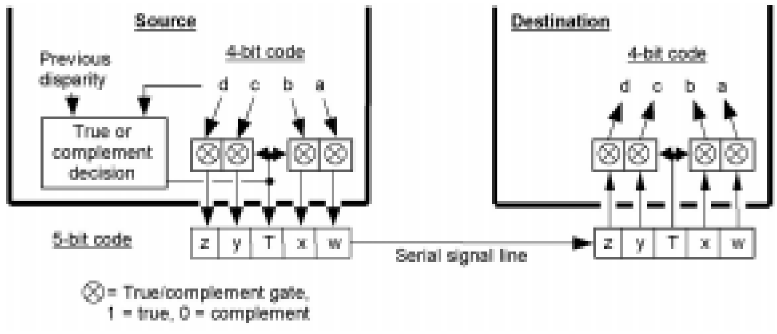

Figure 16.6 4B/5B encoder/decoder.

For example, if the Disparity Count is negative (more 0's than 1's transmitted), and the incoming 4-bit data also has more 0's than 1's, then the incoming 4-bit code is complemented (to generate more 1's), and the "T" bit is set to 0. At the receive end the incoming bits are passed straight through (un-complemented 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, developed by (Widmer and Franaszek, 1983) and covered under a U.S. Patent (Franaszek and Widmer, 1983), 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/5B 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 start in the middle of a 5-bit data pattern, the "T" bit was put on the end. The simulation showed that the operating system had many 4-bit zero patterns (binary '0000'), and these gave 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. Since real user data is more likely to contain '0000' rather than '1111' patterns, this move was also considered useful for the general case.

16.12 DESKEWING THE PARALLEL SIGNALS

The CLOCK signal, used to strobe the other received signals, is carried on a separate line, negating the need for clock recovery circuits on every data line. Up to 10 nanoseconds 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 (40 nanoseconds) every 10 microseconds, accounting for the missing 0.04% of the 6400 Mbit/s total bandwidth. Figure 16.7 is a block diagram of the deskew circuit on one signal line; there are a total of 20 such circuits on an interface chip. Each received signal drives a tapped delay line (implemented as a series of inverters in the SuMAC chip), and the output is selected from one of the taps. A special signal pattern is used to train the deskew logic.

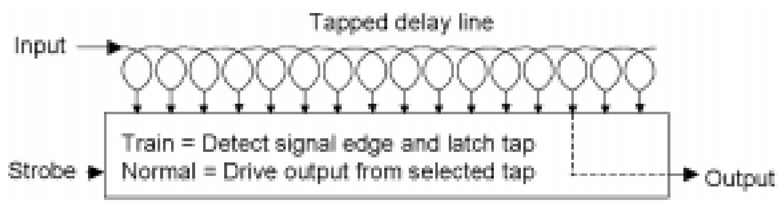
Tapped delay line
Input
Train = Detect signal edge and latch tap
Normal = Drive output from selected tap
Strobe
Output

Figure 16.7 Tapped delay line deskew circuit.

Figure 16.8 shows four signals being deskewed. They are transmitted edge-aligned, but entering the receiver they are skewed due to differences in wire lengths, propagation delay, *etc*. Delay Ckt 1 is adjusted to $\Delta 1$, Delay Ckt 2 to $\Delta 2$, *etc*., so that all of the signals are again edge-aligned as they leave the delay circuits. This implements the deskew function. Not shown is a half-cycle shift of the CLOCK signal so that the CLOCK can sample the other signals in the middle of a bit period.

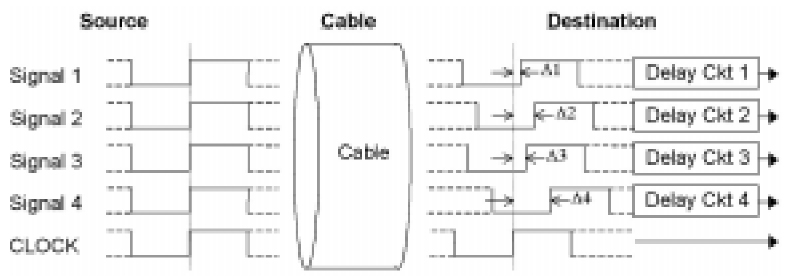

Figure 16.8 Dynamic deskew in operation.

16.13 SUMMARY

HIPPI-6400 is an emerging standard for moving digital data at speeds of up to 6400 Mbit/s (800 Mbytes/s) with very low latency 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.

Acknowledgments

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 the auspices of the U.S. Department of Energy (this paper is LA-UR 97-4906). 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

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

ANSI (1998). The following documents are "draft proposed American National Standard" as of March 16, 1998. The x's in the X3.xxx-199x will be replaced with digits as the documents progress through the ANSI processing.
(1) ANSI NCITS 324-199x, High-Performance Parallel Interface – 6400 Mbit/s Physical Layer (HIPPI-6400-PH).
(2) ANSI NCITS 324-199x, High-Performance Parallel Interface – 6400 Mbit/s Optical Specification (HIPPI-6400-OPT).
(3) ANSI NCITS xxx-199x, High-Performance Parallel Interface – 6400 Mbit/s Physical Switch Control (HIPPI-6400-SC).

Crandall, D., Hessel, S., Hornak, T., Nordby, R., Springer, K., and Walker, R. (1995). Dc-free code for arbitrary data transmission. *U.S. Patent 5438621*.

Franaszek, P. and Widmer, A. (1983). Byte-oriented dc balanced (0,4) 8b/10b partitioned block transmission code. *U.S. Patent 4486739*.

Hoffman, J. (1996). HIPPI-6400: Analysis of a high-throughput network interface. Master's thesis, University of Arizona.

Hornak, T., Lai, B., Petruno, P., Stout, C., Walker, R., Wu, J., and Yen, C. (1991). Dc-free line code and bit and frame synchronization for arbitrary data transmission. *U.S. Patent 5022051*.

Widmer, A. and Franaszek, P. (1983). Dc-balanced, partioned-block, 8b/10b transmission code. *IBM Journal of Research and Development*, 27(5):440–451.