FireWire Brings Fast Serial Bus to Desktop

P1394 Offers Low-Cost 10-Mbyte/s Peripheral Interconnect

by Curtis P. Feigel

A new serial bus championed by Apple promises to simplify the lot of the average computer user. FireWire is a high-performance interconnect developed from IEEE's proposed P1394 standard. It's both low cost and flexible, and it offers speeds scalable from 100 to 400 Mbps, making it a logical replacement for the aging and inconvenient SCSI. Disk drives and digital video cameras are its likely early targets, but this emerging technology will be useful for connecting printers, scanners, and other fast peripherals—and may even replace local-area networks to some degree.

FireWire supports a variety of topologies, allows up to 63 nodes per bus, and eliminates the need for terminators. It is also hot-pluggable: users can add, remove, or rearrange peripherals while a system is running. The system automatically assigns node addresses, so users need not set ID switches. FireWire's protocol also provides guaranteed bandwidth for isochronous data, such as digital video bitstreams that require a constant delivery rate.

Apple demonstrated a working isochronous link at the January MacWorld show, capturing live video on one Quadra machine and transmitting it in real time via FireWire to a second Quadra for display. The system stored video directly onto a hard drive that used a FireWire interface rather than the usual SCSI.

FireWire is not just another way to connect Macintoshes. IBM also demonstrated machines with FireWire interfaces (at the most recent Fall Comdex); Maxtor, Western Digital, and Adaptec are demonstrating a variety of peripherals, and silicon is due from both Texas Instruments and NCR.

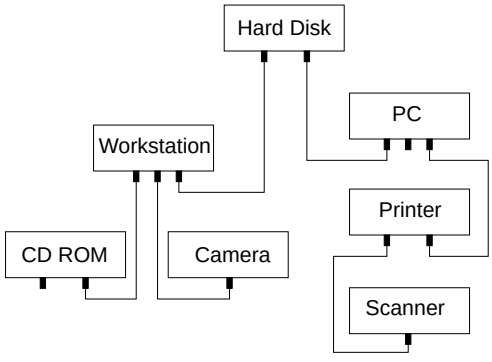

Figure 1. FireWire's point-to-point connections allow combined daisy-chain and tree topologies with a maximum of 16 hops. Current performance levels limit node-to-node distance to 4.5 meters.

Open Standard Solves Problem

SCSI's less friendly qualities provided the impetus for the FireWire project. According to Apple, 40–50% of its support calls are related to SCSI cabling, addressing, and termination problems. Apple recognized that modern IC processes could build a serial interface that was as fast as or faster than the current SCSI and that the resulting overall solution—including ICs, cables, and connectors—would be less expensive.

Apple chose the IEEE's proposed standard P1394 and implemented it, calling the resulting product FireWire. The Verilog design was then licensed to TI and NCR—companies that had the expertise to solve mixed-signal issues and could produce the FireWire chip set as a high-volume commercial product. Because it's developed from this open standard, interested third parties can create FireWire peripherals and interface chips that are interoperable.

By design, P1394 follows conventions developed for the IEEE 1212 standard, which specifies control- and status-register architecture. This standard is also used in SCI (Scalable Coherent Interface, IEEE 1596) and Futurebus (IEEE 896) and provides a common basis for such things as addressing, error status codes, node IDs, and read, write, and lock transactions. Although FireWire is implemented for a cable, P1394 also allows the serial bus to be implemented using a few pins of a backplane.

What the user sees is a new six-wire round cable that is small, flexible, and inexpensive yet rugged (the connectors are rated for up to 5,000 insertion cycles). This cable connects peripherals, processors, and other nodes that have two or more FireWire ports. Its "plug and play" operation eliminates concerns about switch settings, termination, and even the order in which devices are connected.

As shown in Figure 1, the connections are point-to-point. Combinations of daisy-chains and tree-like structures are allowed, but not loops. (Users can't make a loop if they have n nodes and only n-1 cables, so manufacturers plan to bundle cables only with peripherals and not with host systems.) Nodes pass along data intended for other targets. The scheme allows for up to 16 of these "hops" in any path.

Initial controllers will support the base rate of approximately 100 Mbps (98.304 Mbps—developed from a frequency common in the telecommunications industry). But the standard also allows speeds that are two and

four times the base rate to coexist on the same bus. Nodes that support speeds higher than the base rate are required to also support all lower speeds.

The standard does not set a maximum internode distance other than by performance. TI's first controller implementations will allow cables up to 4.5 meters, but the cable length can increase as the logic becomes faster.

Model Is a Memory Map

Despite its serial nature, FireWire operates very much like a bus—at least from the programmer's point of view. Each node is memory-mapped into its own segment of a 64-bit address space, allocated as Figure 2 shows. Six of these bits address up to 63 nodes on a bus; addressing the 64th node causes a broadcast to all nodes on that bus. This scheme also supports systems made of multiple buses bridged together: 10 of the bits extend the reach across as many as 1023 buses, for a total of 64,512 nodes (addressing the 1024th bus automatically accesses the local bus).

This leaves 48 bits for addresses within each node. Each address specifies a byte within a 32-bit word, so every node has a vast 256-terabyte (262,144-gigabyte) address space! This is allocated as follows:

- Initial Memory Space—A 255.5-terabyte window that allows access to a node as if it were memory
- Private Space—256M allocated for vendor-dependent resources local to the node, not seen on the bus
- Control and Status Registers—512 bytes allocated for IEEE 1212-type CSRs
- Serial Bus—512 bytes allocated for registers specific to P1394 functions
- ROM Space—1K to support either of two ROM formats: minimal (used for inexpensive nodes) and general (allows pointers)
- Initial Units Space—256M allocated for node-specific resources such as processors and I/O

Figure 3 shows one view of the protocol stack. In the TI chip set, the protocols are implemented in a combination of hardware and firmware controlled by the set of standard CSRs in the serial bus management block. The transaction layer provides a complete interface for requests from and responses to the application software or operating system. The link layer assembles and disassembles data packets and provides acknowledgements to the transaction layer. Isochronous data must be passed directly from the application to the link layer, although the transaction layer is used to manage isochronous transfers. The physical layer arbitrates and translates the packets to and from the electrical signals on the serial bus.

At the physical level, each FireWire port uses two bidirectional, differential signals to carry clocked packets of data and arbitration information. In the FireWire cable, these occupy two separate shielded twisted pairs with a controlled impedance of 110 Ω. The cable is built to swap the two pairs, so the data transmitted on pair A of one node is received on pair B of the other node, and vice versa. The low-voltage, low-current signals swing in a very small range from 170 to 260 mV, reducing transition times. Each node may also source or sink power on a third pair of low-resistance (less than 1 Ω) wires that carry up to 1.5 A.

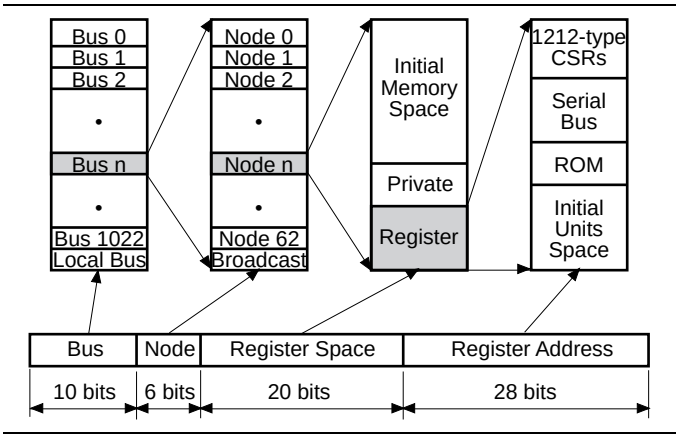

Figure 2. P1394 draws upon the IEEE 1212 standard for its definition of 64-bit fixed addressing. Sixty-three nodes on 1024 buses are each assigned 256 terabytes of address space.

Within the TI chip set are separate sets of drivers and receivers for the data signals and the arbitration signals. The link-layer chip contains a protocol engine occupying 5,000 to 10,000 gates; the rest is devoted to FIFOs, DMA, and the bus interface.

Fast Configuration Allows Hot-Plugging

Because it allows up to 63 nodes, the arrangement of a FireWire bus can be quite complicated. Florin Oprescu, then of Apple, developed a dynamic tree-building algorithm that the nodes use to discover and map

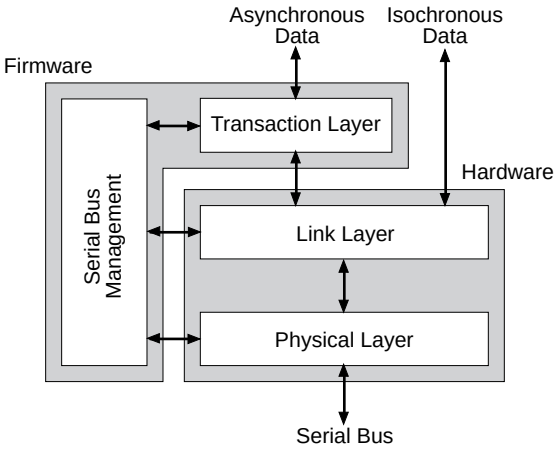

Figure 3. The TI chip set implements layers of the protocol stack partially in hardware, partially in firmware.

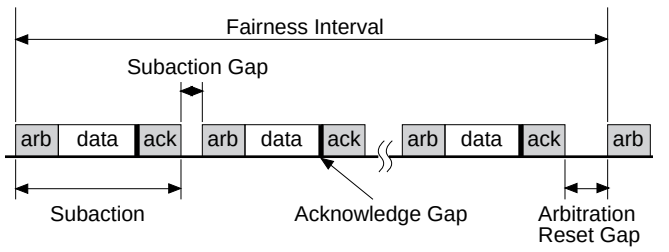

Figure 4. Nodes acknowledging transactions receive higher priority and are allowed to transmit immediately after any data packet. Other nodes must wait longer to arbitrate for bus ownership, creating the subaction gap.

their topology. This is done in three phases: bus initialization, tree ID, and self ID.

Bus initialization clears all previous topology information. At this point, each node knows only whether it is a leaf (connected to only one other node) or a branch (connected to more than one node). The tree ID process works its way inward from the leaves until one node is identified as the root. Then every node labels each of its ports as a parent (towards the root), a child (away from the root), or unconnected. Finally, during the self ID phase, each node selects a unique physical ID and identifies itself to any management entity attached to the bus, describing its ports and their speeds. Software in any node can then construct a map of the bus showing what path data must take to get to any other node.

When a new device joins the bus, pending transactions are completed, the old topology is discarded, and the bus is reconfigured to admit the new node. Worst-case reconfiguration requires 400 µs (about the time it takes to display 10 rows of pixels on a VGA monitor). This is easily fast enough to allow hot-plugging of peripherals in a running system. All bus-management communications take place at the base rate.

Split Transactions Are Robust

Transmissions over the bus follow a datagram model (data is accompanied by an address), with split transactions consisting of a request and a later response. To ensure robust communications, each request or response on the bus is acknowledged with a single-byte packet so the originating node knows the link is working. The node will retry the transaction if it does not receive the expected response. This part of the protocol is handled by the link layers of the two nodes, and each complete request/acknowledge or response/acknowledge cycle is called a subaction.

To obtain data from a particular node, the application commands the transaction layer to request a read, specifying the address and length of the data. The link layer processes the request, sends a confirmation back to the transaction layer (so it can resume its operation), and passes the request across the bus via the physical layer. Once the request is routed to the target node, it is processed by that node's link layer, which sends an acknowledgment to the originating node, completing the subaction. The target node passes the information to its transaction layer, which announces the arrival of a read request to the application.

The response subaction occurs once the application at the target node has completed the read operation. The data is passed through the node's transaction layer, where it is assembled into a packet that also contains the data length, the address of the source node, and a response code. This packet is passed through the link layer and is routed across the bus to the originating node, where the link layer returns an acknowledgment to the target, completing the response subaction. The various layers within the originating node perform their functions, completing the whole transaction.

Writing data to a particular node follows a similar series of actions, except that the initial request contains the data to be written, while the response contains a simple code indicating the success or failure of the operation. Because the serial bus operates predominantly in a split-transaction mode, the application layer can request several types of special locked transactions that permit indivisible test-and-set operations. The system also supports broadcast and retry transactions. The transaction layer can track up to 63 pending transfers at a time.

Arbitration Fits Between Data Packets

Priority of ownership of the bus is based on gaps between data traffic, as shown in Figure 4, and the fact that data and arbitration signals go over separate wires. Nodes sending acknowledgments achieve the highest priority because they do not need to arbitrate; instead, each listens on the bus for a short interval called an acknowledge gap. If the bus is not busy with data traffic, the node can begin transmitting. Nodes sending isochronous data have next priority and try to gain bus ownership by asserting the arbitration signal while the data bus is idle.

Nodes sending asynchronous data must wait for the longer subaction gap between request or response subactions (each of which may be immediately followed by an acknowledge packet). As each of these nodes wins bus ownership, its Arbitration Enable bit is cleared, preventing it from arbitrating again. Eventually, none of these nodes can arbitrate, and the interval between subactions stretches into a long gap that resets the Arbitration Enable bits of all nodes, and the process begins again. This defines fairness intervals during which each arbitrating node can win only once.

Nodes send arbitration requests to their parents in the tree structure. The requests are passed up the tree to the root node. The root grants access to the first fair request it receives and denies the others.

Price & Availability

TI's FireWire chip set should be available during 1H94 for between $20 and $40. For more information, contact Texas Instruments at 214.997.3426. For draft copies of the proposed IEEE standard P1394 "High Performance Serial Bus," contact Global Engineering Documents at 800.624.3974.

Packet Size Is Based on Performance

The P1394 document defines 10 different asynchronous-data packets, but in general they consist of a sequence of aligned long words beginning with a header, ending with a cyclic redundancy check, and carrying a variable-length data block. The standard limits the size of a data block by the amount of time required to send it. Thus, the maximum data block is 512 bytes at 100 Mbps, 1 Kbyte at 200 Mbps, and 2 Kbytes at 400 Mbps.

Isochronous-data packets are similar in structure but have a simplified header and a maximum data-block size twice that of the asynchronous packets. These changes reduce the administrative overhead and allow higher throughput for time-sensitive data.

Designers and programmers will be relieved to hear that these protocol issues are hidden, so they need to be concerned only with the application-layer interface. All this overhead does raise the question of latency. The worst-case conditions are complex, but simulations show that the time required to repeat data through TI's physical-layer chip is 144 ns. Asynchronous data through seven hops of a FireWire bus would have a latency of about one microsecond, allowing about one million transactions per second.

Industry Shows Support

FireWire appears to have support in all the right places: system makers Apple and IBM as well as disk-related manufacturers Adaptec, Maxtor, NCR, and Western Digital have all demonstrated interest by building and showing hardware. General-purpose silicon is on its way from TI in the form of a link-layer and physical-layer chip set costing $20 to $40—awaiting only approval of the standard (perhaps during March).

NCR is working on three related devices: a PCI-to-FireWire bridge, a physical-layer chip that supports the 200-Mbps speed (both to be introduced around midyear), and a link-layer chip destined for target peripherals such as printers and scanners (4Q94). The company has indicated it will make its link-layer solution available as a drop-in macrocell for use in ASICs.

Initially, companies will buy off-the-shelf chip sets, but it's likely that computer makers such as Apple and IBM will incorporate the link-layer interface into their ASICs while retaining the off-the-shelf physical-layer chip. Pricing pressure will drive embedded applications, such as hard disks and video cameras, to incorporate both chips into their own silicon.

	FireWire	SCSI-2	FDDI	RS232
Maximum Number of Nodes	63	16	1024	2
Peak Bandwidth (Mbytes/s)	12.5–50	20	12.5	0.01
Node-to-Node Cost	$40	$70	$700	$5.50
Cost per Mbyte/s	$3.20	$3.75	$56	$550
Self Configuring?	Yes	No	Yes	No
Terminators Required?	No	Yes	No	No

Table 1. FireWire allows speeds equivalent to or better than other contenders while maintaining low cost and using simpler cabling. (Sources: Apple and NCR)

For its cost, FireWire offers significant bandwidth. It's fast enough to support hard-disk speeds that are a step or two beyond today's drives. And considering the current rate of progress in building mixed-signal ICs, it's conceivable that the standard could be extended beyond 400 Mbps.

FireWire has advantages over each of the competitors shown in Table 1. FDDI has similar speed, and its optical-fiber cabling allows longer links, but its cost per node is much greater. Wide SCSI-2 may be faster than P1394's base rate, but wide SCSI hard drives and controllers are still rare. And SCSI's termination, addressing, and cabling make the average user cringe. Also, both TI and NCR are working on faster FireWire chip sets. The decades-old RS232 standard, although slower, is well established, and its lower cost guarantees it will find a place in future machines. Other schemes such as ATM, Fiber Channel, and IBM's SSA are universally more expensive and less available.

SCSI will likely be the first of the interfaces to fall to FireWire, considering that four of FireWire's announced backers are in the business of manufacturing disk drives or controllers. IBM's support hints that PCs, which have not employed SCSI to the degree the Macintosh has, could also gain flexibility from FireWire.

Manufacturers are also looking at consolidation of I/O ports—connectors, and even access holes in cases, cost money. Considering the current trend toward multimedia-type I/O in personal computers, FireWire also offers economy. Multimedia-equipped computers require numerous extra ports, each with its own interface and drive circuitry; much of this can be replaced with a single FireWire port, assuming video devices also migrate to the standard.

Optimists in the FireWire camp foresee the day when all a computer's peripherals, even its keyboard and mouse, are connected via this serial bus. It may not happen quite this way, but FireWire is compelling because it's both inexpensive and fast. From the user's point of view, FireWire is an elegantly simple technology. Its plug-and-play philosophy will streamline the way we use computers. ♦

