

IEEE 1394 compared with SpaceWire

In many respects, [SpaceWire](#) and 1394 are very similar. They are both serial, they both have data rates around 100Mbits/s, they both use Data/Strobe encoding, they both use small vertically mounted connectors, they both use point-to-point cables, they are both designed for short-distance interconnect rather than a global network.

There is one fundamental difference. 1394 is a logical bus, whereas SpaceWire is designed for simple packet switching through hardware switch fabric.

In other areas, the bus has been discredited. ATM uses switch fabric. FibreChannel uses switch fabric. Myrinet uses switch fabric. New installations of Ethernet use switch fabric. PCI Express uses switch fabric. Advanced TCA uses switch fabric.

The reason for the bus being discredited is simple. As you add more nodes, you need more performance. But what actually happens when you add nodes is that the bus delivers less. The bus is a bottleneck.

It was always considered that switch fabric was expensive. It was indeed expensive when integration levels were SSI, when the silicon was slow, and when the protocols were complex. But the advance in silicon integration levels and performance make such reasoning obsolete.

There may still be a place for the bus, as long as it is inside a silicon chip. A 64-bit bus in a 1GHz CPU has a bit-rate of 64Gbits/s --- and there can be plenty of such buses inside a chip. So the potential bit-rate inside a modern chip can even approach 1Tbit/s --- many orders of magnitude greater than is conceivable on a cabled bus.

Even a much lower bit-rate inside the silicon can produce highly effective switch fabric. It helps also if the port logic is simple, so that either more ports can be used on one chip or so that a small chip can be used. In 4Links' switch fabric implementations in Xilinx FPGA, we have an internal bit rate of about 400Mbits/s per port. So for an eight-port routing switch, we have an internal bit-rate of 3.2Gbits/s. Each port is full-duplex, so we can claim a device bit-rate of 6.4Gbits/s. If more throughput is needed, we use a bigger chip with more ports, or we use several such chips connected together.

Meanwhile 1394 finds 100Mbits/s to be inadequate, so goes to 400Mbits/s. And then finds that is still inadequate and so goes to 800Mbits/s and faster. Each of these increases in speed means that the overhead of arbitration uses a higher and higher proportion of the available bit-rate, and so the faster bit-rates produce diminishing returns. And if there is one slow device on the bus, all the fast devices have to slow down for it. And yet the fastest bit-rates for 1394 are significantly lower than SpaceWire can achieve (several Gbits/s) in a single switch-fabric chip based on a cheap FPGA.

So the principal problem with 1394 is that it was obsolete before it arrived.

There are other problems with 1394.

4Links

The topology for 1394 is required to be a tree or a star, or a combination of the two. Break any node or any link in a tree and the tree splits into two trees that can not communicate with each other. Redundant connections, which could offer an alternative path to fix the break, would form a loop, and 1394 prohibits loops. So *any* fault, anywhere on the bus, breaks the whole bus. Not nice for home networks, disastrous for space.

The default communication model of 1394 is of a memory bus, with any node on the bus able to write to any location in any other node on the bus. Put another way, any node on the bus can corrupt any other node.

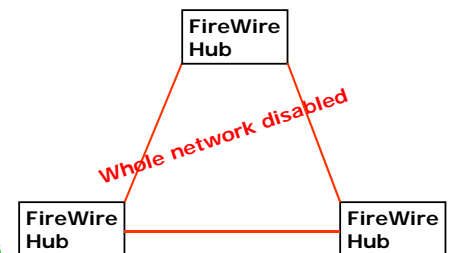
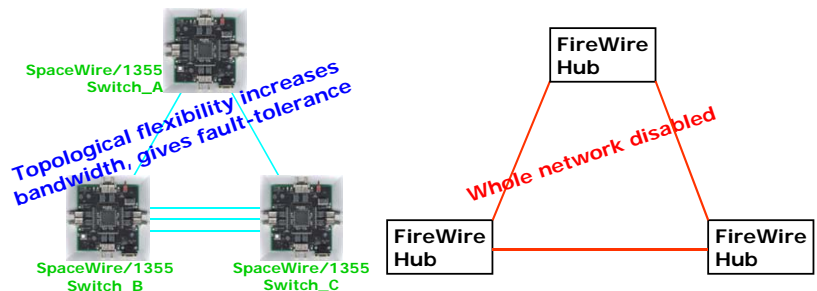
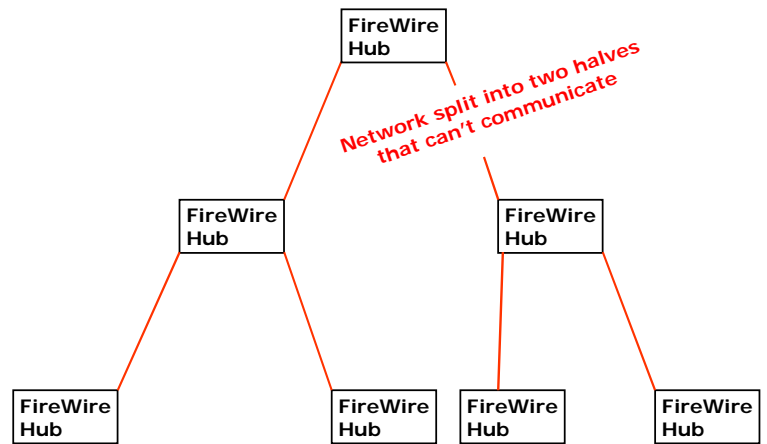
The original 1394-1995 was complex, but the complexity might have been manageable. With 1394a and 1394b and 1394.1, all interacting in subtle ways, the complexity has become unmanageable. Complexity, of course, turns into costs, into incompatibilities from different interpretations, into bugs, and into project delays. Just considering costs, 4Links have fitted eight physical ports for SpaceWire and an eight-port routing switch into a 300kgate FPGA. The smallest FPGA we've seen for a single 1394 port is 1M gates, and it still needs external PHY chips.

Another major problem with 1394 was illustrated in the standardization process. The original PAR (Project Authorization Request) for 1394a included the following three items:

- c A connection management protocol, and the necessary CSR facilities, for isochronous channels
- e An incremental reconfiguration protocol, invoked when a device is connected to or disconnected from Serial Bus, which is designed to limit the circumstances under which a Serial Bus reset is necessary
- f a technique for Serial Bus topologies that from a physical loop to be resolved into a logical tree structure (required by existing standard) without user intervention.

These are all clear requirements, and the PAR was correct to include them. They were difficult, however, and the Working Group was unable to propose solutions. So their solution was to delete these necessary items from the project scope and it was this weakened scope that became the revised standard. SpaceWire can, as shown above, be connected in loops (item f), it automatically reconnects links without needing any sort of global reset (item e), and 4Links provides a connection management protocol that impresses all who see it (item c).

It may be that nothing is perfect and everyone could always do better next time. But the points raised seem to be fundamental flaws in 1394. By comparison, SpaceWire, is simple, flexible, robust, and scales with growth of the network. To some extent SpaceWire offers a new paradigm of cheap packet switching through hardware switch fabric. That paradigm is supported by the trend shown by ATM, FibreChannel, switched Ethernet, and PCI-Express, but SpaceWire surpasses all of these in its implementation cost and flexibility.



4Links

A useful analogy to this paradigm shift has already occurred in manufacturing over the last fifty years. There are strong parallels between 1394 and the old-fashioned production line and between SpaceWire and the new flexible just-in-time manufacturing facility. They show that a shift to the switch fabric paradigm of SpaceWire, at some time, is inevitable. That time is now.

Comparison between 1394 and SpaceWire based on analogy with modern manufacturing

Old production line	IEEE 1394	Modern manufacturing	SpaceWire
Bottlenecks	Logical bus means only one transaction can take place at once, creating huge bottleneck	Spare or flexible capacity to eliminate bottlenecks	Multiple paths and adaptive routing to eliminate bottlenecks, provide fault tolerance
Rigid production lines	Topology MUST be a tree	Flexible manufacturing	Flexible encapsulation of other protocols, flexible network topologies
Quality doesn't matter/repair if faulty	Errors are expected and cause retry	Quality is paramount, Zero defects	Reliable links, buffers never overflow
Don't change it	More nodes always need more throughput, always result in less throughput	Continuous improvement	Adding nodes and links continuously improves the network
Lead time is irrelevant	All senders have to queue up to have access to the bus, causing long latency. Even isochronous traffic has 125 us jitter on delay.	Just-In-Time because delay reduces quality, reduces customer service, and reduces throughput	Worm-hole routing and flow-control minimize delay
Huge overheads	Large overheads require large packets to keep up utilization	Minimize overheads	Packet header can be just one byte. Overheads are appropriate for application
Long conveyor belt	One big bus	Many small, independent, cells	Many independent links
Large batch sizes	Large async packets, Isoch packets can be small, but have excessive overhead	Small batches, in trays, preferably one unit per tray	Small packets, appropriate for the application
"Push" components onto the line	No flow-control	When a tray is used, the tray goes back as an order to "pull" more components	Feedback flow-control "pulls" data when there is space for it
Long change-over time	Single arbitration across whole bus.	Very fast tool-change	Fast, widely distributed, independent, arbitrations; fast changeover between different packets and protocols
Environment does not matter	Data visits every node, even if not required by the node	Waste of any kind is damaging to the environment (as well as profit)	Eliminate wasted data from buffer overflow, eliminate wasted power driving unnecessary wires
Inventory is asset	Large buffers required	Inventory is liability	Minimize data buffers, unless application requires buffers
Make 50% profit margin once per year	Shared medium requires 800 Mbps and beyond, even though throughput is limited by slowest device on the bus	Make 20% profit margin 10 times per year	1/5th the link speed in a switched network can offer 20 to 200 times the overall throughput of the bus

[Paul Walker, 4Links Limited](#), Oct. 2001, revised Dec. 2002, May 2004 and Jan 2006. The manufacturing analogy was first published in the author's guest editorial to [Microprocessors and Microsystems](#), Vol 21, nos. 7-8, 30 March 1998.