Fundamentals of Firewire™

Questra CONSULTING
SOFTWARE & EMBEDDED
SYSTEMS GROUP

Author: John Canosa

Questra CONSULTING

Introduction

We have all heard about the coming convergence of computers, consumer equipment and communications. There are many factors driving this convergence. The incredible amount of processing power available for ridiculously low costs and the vast amounts of information that is already available, with more being generated every day, are two driving forces. But communications is the real force that is drawing these separate market segments together.

A larger and larger portion of the new information being generated today is taking the form of multimedia. Video, still images and audio are becoming ubiquitous and causing an increasing thirst for easier, faster ways of transferring information. Convergence will happen when seamless, high speed communications becomes readily available. The IEEE 1394 protocol appears to be a strong contender for the communications channel that will make this happen.

The IEEE 1394-1995 protocol had its genesis at Apple Computer, which still retains the Firewire trademark. The goals of the protocol are to provide easy to use, low cost, high speed communications. In addition, the protocol is very scalable, provides for both asynchronous and isochronous applications, allows for access to vast amounts of memory mapped address space and, perhaps most important to convergence, allows peer-to-peer communication.

Some people see 1394 and USB as competitors for the communications channel of the future, but in reality they are more complementary that competitive. USB is a lower speed, lower cost, host based protocol. While 1394 and USB may compete in some mid-range applications, Figure 1shows that they will typically play in different spaces.

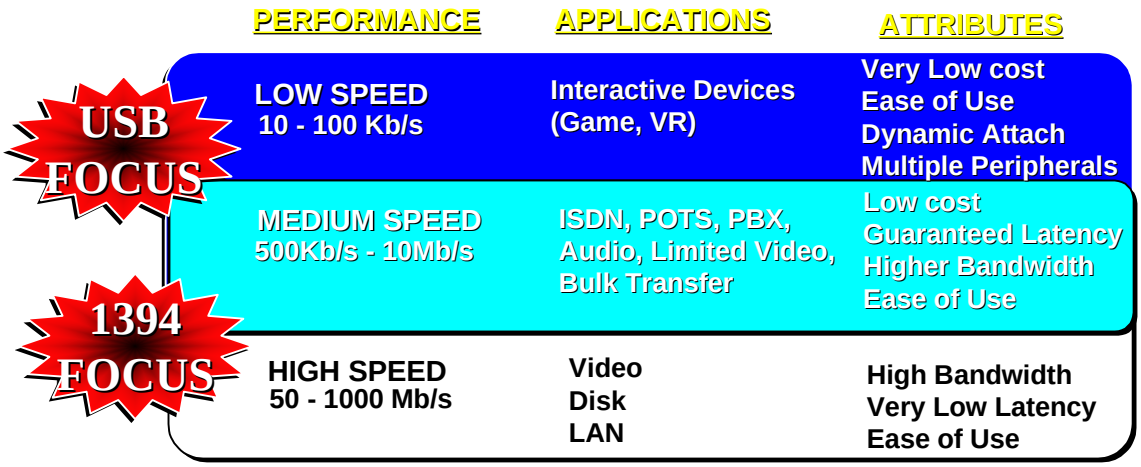

Figure 1 IEEE 1394 and USB Market Segmentation

There is sometimes confusion surrounding the alphabet soup that seems to envelop the 1394 protocol. The only currently approved specification is the IEEE 1394-1995 specification. This specification is the basis for future extensions and enhancements. 1394-1995 supports transfer rates of 100, 200 and 400 Mbps. As with many first cuts at a standard, 1394-1995 left some things up to interpretation of the implementers of the specification. This caused some interoperability problems and has led to work on the 1394a specification. The 1394a specification provides some clarification on the original specification, changes some optional portions of the spec. to mandatory

Questra CONSULTING

and also adds some performance enhancements. The 1394a specification is nearing completion and should be approved in the near future – some semiconductor vendors are already claiming compliance to the new specification. In addition to the 1394a specification, work is progressing on the 1394b specification. 1394b will provide for additional data rates of 800, 1600 and 3200 Mbps. It will also provide for long haul transmissions via both twisted pair and fiber optics as well as providing backward compatibility with the existing standard. There also had been work done by Intel on a protocol called 1394.2. This also was to provide higher speed communications but was not backward compatible with the existing specification. Publicly, this work has since been merged with 1394b development, but who knows what hot new technology might pop out of the Intel Architecture Labs in the future. This article will cover the 1394-1995 standard and will speak to some of the enhancements in the 1394a revision. Details of the 1394b protocol will be left for a future article, when the specification is more firm.

The Topology

The 1394 protocol is a peer-to-peer network with a point to point signaling environment. Nodes on the bus may have several ports on them, each of these ports act as repeaters, retransmitting any packets that are received by other ports within the node. Figure 2 shows what a typical consumer may have attached to their 1394 bus.

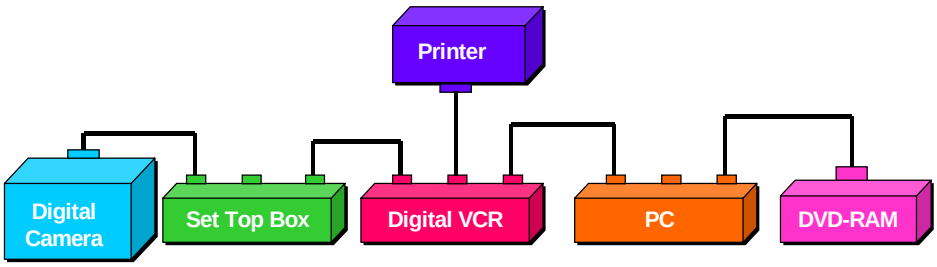

Figure 2 A Firewire Bus

Because 1394 is a peer to peer implementation, there is not a specific host required, such as a PC in USB. In the above figure, the Digital Camera could easily stream data to both the Digital VCR and the DVD-RAM without any assistance from other devices on the bus. There is the concept of a Cycle Master and Root Node, which is discussed subsequently. The Cycle Master function does not require much added sophistication. The determination of what device plays the Cycle Master role is determined dynamically and can change whenever a new device is added or a bus reset occurs.

Configuration of the bus occurs automatically whenever a new device is plugged in. Configuration proceeds from leaf nodes (those with only one other device attached to them) up through the branch nodes. A bus that has three or more devices attached will typically, but not always, have a branch node become the Root Node. Configuration will be discussed in more detail later in this article.

A 1394 bus appears as a large memory mapped space with each node occupying a certain address range. The memory space is based to the IEEE 1212 Control and Status Register (CSR) Architecture with some extensions specific to the 1394 standard. Each node supports up to 48 bits of address space (256 TeraBytes). In addition, each bus can support up to 64 nodes and the 1394 serial bus specification supports up to 1024 busses. This gives a grand total of 64 address bits, or support for a whopping total of 16 ExaBytes of memory space – enough for the latest version of your favorite word processor and perhaps even a file or two!

Questra CONSULTING

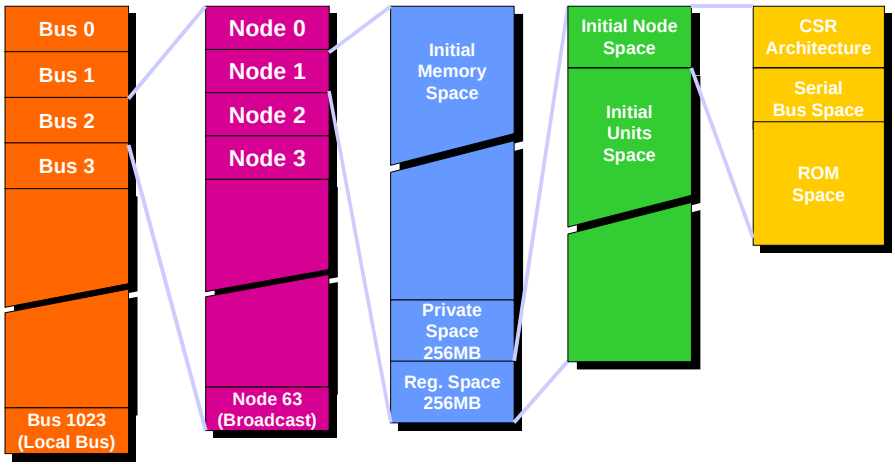

Figure 3 IEEE 1394 Address Space

Transfers and Transactions

Isochronous and Asynchronous Transfers

The 1394 protocol supports both asynchronous and isochronous data transfers. Isochronous transfers are always broadcast in a one to one or one to many fashion. There is no error correction or retransmission available for isochronous transfers. Up to 80% of the available bus bandwidth can be used for isochronous transfers. The delegation of bandwidth is tracked by a node on the bus that is occupying the role of Isochronous Resource Manager. This may or may not be the Root node or the Bus Manager. The maximum amount of bandwidth an isochronous device can obtain is only limited by the number of other isochronous devices that have already obtained bandwidth from the isochronous resource manager.

Asynchronous transfers are targeted to a specific node with an explicit address. Asynchronous transfers are not guaranteed a specific amount of bandwidth on the bus, but they are guaranteed a fair shot at gaining access to the bus when asynchronous transfers are allowed. The maximum data block size for an asynchronous packet is determined by the transfer rate of the device as shown in the following table.

Cable Speed	Maximum Data Size
100 Mbps	512 Bytes
200 Mbps	1024 Bytes
400 Mbps	2048 Bytes

Table 1

Asynchronous transfers are acknowledged and responded to. This allows error checking and retransmission mechanisms to take place. The bottom line is that if you are sending time critical, error tolerant data, such as a video or audio stream, then isochronous transfers are the way to go. If the data is not error tolerant, such as a disk drive, then asynchronous transfers are preferable.

Questra CONSULTING

Four Layers of Protocol

The 1394 specification defines four protocol layers, although not all of them are used during all transfers. The layers are shown in Figure 4.

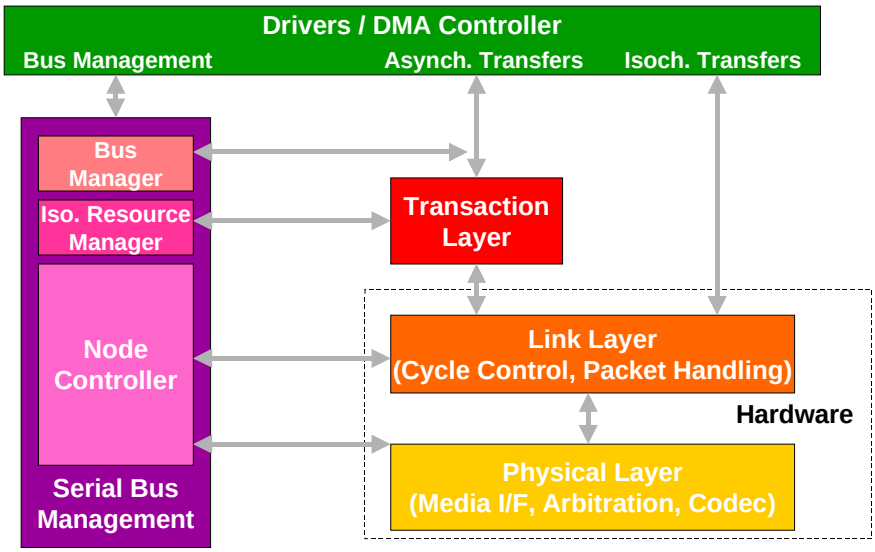

Figure 4 IEEE 1394 Protocol Layers

Physical Layer

The physical layer of the 1394 protocol includes the electrical signaling, the mechanical connectors and cabling, the arbitration mechanisms and the serial coding and decoding of the data being transferred or received. The cable media is defined as a three pair shielded cable. Two of the pairs are used to transfer data, while the third pair provides power on the bus. The connectors are small six pin devices, although the 1394a also defines a 4 pin connector for self powered leaf nodes. The power signals are not provided on the 4 pin connector. The baseline cables are limited to 4.5 meters in length. Thicker cables allow for longer distances.

The two twisted pair used for signaling, called out as TPA and TPB, are both bidirectional and are tri-state capable. TPA is used to transmit the strobe signal and receive data, while TPB is used to receive the strobe signal and transmit data. The signaling mechanism uses data strobe encoding, a rather clever technique that allows easy extraction of a clock signal with much better jitter tolerance than a standard clock/data mechanism. With Data Strobe encoding, either the data or the strobe signal change in a bit cell, but not both of them. Data Strobe encoding is shown in Figure 5.

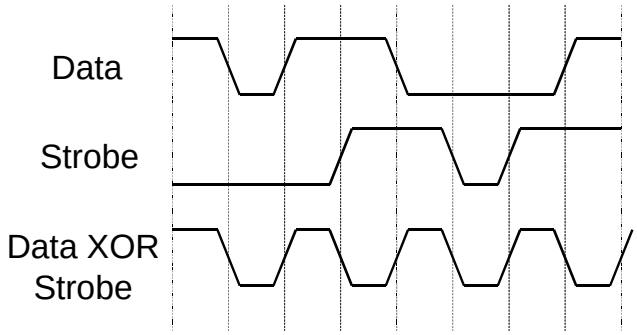

Figure 5 Data Strobe Encoding

Configuration

The physical layer also plays a major role in the bus configuration and normal arbitration phases of the protocol. Configuration consists of taking a relatively flat physical topology and turning it into logical tree structure with a root node at its focal point. A bus is reset and reconfigured whenever a device is added or removed – a reset can also be initiated via software. Configuration consists of bus reset and initialization, tree identification and self identification.

Reset

Reset is signaled by a node driving both TPA and TPB to logic 1. Because of the "dominant 1s"electrical definition of the drivers, a logic "1" will always be detected by a port, even if its bidirectional driver is in the transmit state. When a node detects a reset condition on its drivers, it will propagate this signal to all of the other ports that this node supports. The node then enters the idle state for a given period of time to allow the reset indication to propagate to all other nodes on the bus. Reset clears any topology information within the node.

Tree Identification

The tree identification process is how the bus topology is defined. Lets take the example of our sample home consumer network shown in Figure 2. After reset, but before tree identification, the bus has a flat logical topology that maps directly to the physical topology. After tree identification is complete, a single node has gained the status of Root Node. The tree identification proceeds as follows:

After reset, all leaf nodes (nodes with only one other device connected to them) present a Parent_Notify signaling state on its Data and Strobe pairs. Note that this is a signaling state, not a transmitted packet – the whole tree identification process occurs in a matter of microseconds. In our example, the Digital Camera will signal the Set-Top Box, the Printer will signal the Digital VCR and the DVD-RAM will signal the PC. When a branch node receives the a Parent_Notify signal on one of its ports, it marks that port as containing a child, and outputs a Child_Notify signaling state on that port's Data and Strobe pairs. Upon detecting this state, the leaf node marks its port as a parent port and removes the signaling, thereby confirming that the leaf node has accepted the child designation. At this point our bus appears as shown in Figure 6. The ports marked with a P indicates that a device that is closer to the Root Node is attached to that port, while a port marked with a C indicates that a node further away from the Root Node is attached. The port numbers are arbitrarily assigned during design of the device and play an important part in the self identification process.

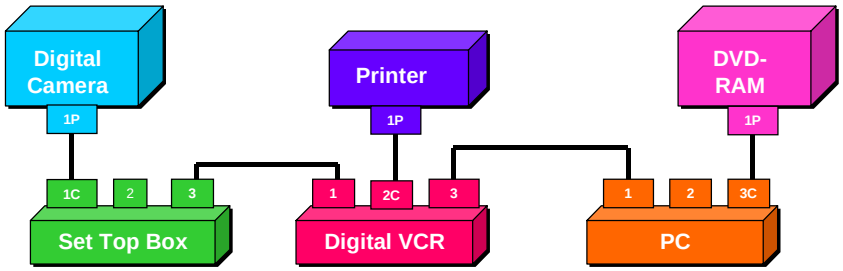

Figure 6 Bus After Leaf Node Identification

After the leaf nodes have identified themselves, the Digital VCR still has 2 ports that have not received a Parent_Notify, while the Set Top Box and the PC branch node both have only one port with an attached device that has not received a Parent_Notify. This being the case, both the Set Top Box and the PC start to signal a Parent_Notify on the one port that has not yet received one. In this case the VCR receives the Parent notify on both of its remaining ports, which it acknowledges with a Child_Notify condition. Since the VCR has marked all of its ports as children, the VCR becomes the Root Node. The final configuration is shown in Figure 7.

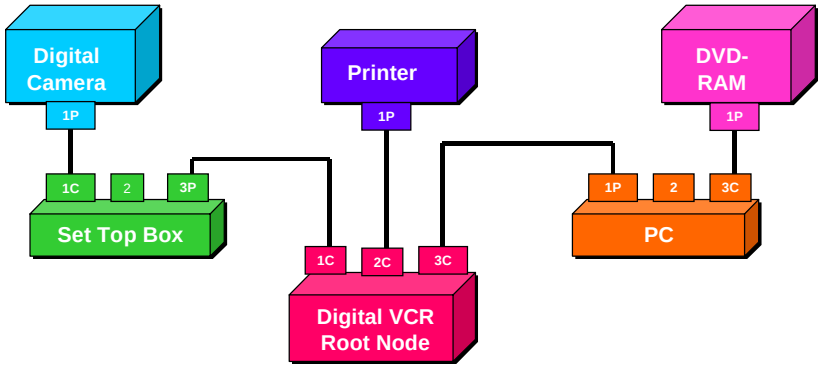

Figure 7 Bus After Tree Identification Is Complete

Note that it is possible for two nodes to be in contention for Root Node status at the end of the process, in this case, a random back-off timer is used to eventually settle on a Root Node. It is also possible for a node to force itself to become Root Node by delaying its participation in the Tree Identification process for a while. Refer to references 1 and 2 for more details.

Self Identification

Once the Tree topology is defined, the Self Identification phase begins. Self Identification consists of assigning Physical IDs to each node on the bus, having neighboring nodes exchange transmission speed capabilities and making all of the nodes on the bus aware of the topology that exists. The self identification phase begins with the Root Node sending an Arbitration Grant signal to its lowest numbered port. In our example the Digital VCR is the Root Node and it signals the Set Top Box. Since the Set Top Box is a branch node, it will propagate the Arbitration Grant signal to its lowest numbered port with a Child node attached. In this case this is the Digital Camera. Since the Digital Camera is a leaf node, it cannot propagate the Arbitration Grant signal downstream any further, so it assigns itself Physical ID 0 and transmits a Self-ID packet upstream. The branch node (Set Top Box) repeats the Self ID packet to all of its ports with attached devices. Eventually the Self ID packet makes it way back up to the Root Node, which proceeds to transmit the Self ID packet down to all devices on its higher numbered

ports. In this manner, all attached devices receive the Self ID packet that was transmitted by the Digital Camera. Upon receiving this packet all of the other devices increment their Self ID counter. The Digital Camera will then signal a Self ID Done indication upstream to the Set Top Box. This indicates to the Set Top Box that all nodes attached downstream on this port have gone through the Self ID process. Note that the Set Top Box does NOT propagate this signal upstream toward the Root Node because it has not completed the Self ID process.

The Root Node will then continue to signal an Arbitration Grant signal to its lowest numbered port which in this case is still the Set Top Box. Since the Set Top Box has no other attached devices, it assigns itself Physical ID 1 and transmits a Self ID packet back upstream. This process continues until all ports on the Root Node have indicated a Self ID Done condition. The Root Node the assigns itself the next Physical ID – the Root Node will always be the highest numbered device on the bus. If we follow our example through we come up with the following Physical Ids – Digital Camera = 0, Set Top Box = 1, Printer = 2, DVD-RAM = 3, PC = 4 and the Digital VCR, which is the Root Node is assigned Physical ID 5.

Note that during the Self ID process, parent and children nodes are also exchanging their maximum speed capabilities. This process also exposes the Achilles' Heel of the 1394 protocol. Nodes can only transmit as fast as the slowest device in between the transmitting node and the receiving node. For example, if the Digital Camera and the Digital VCR are both capable of transmitting at 400 Mbps, but the Set Top Box is only capable of transmitting at 100Mbps, there is no way for the high speed devices to use the maximum rate to communicate amongst themselves. The only way around this problem is for the end user to reconfigure the cabling so the low speed Set Top Box is not physically in between the two high speed devices.

Also during the Self ID process, all nodes wishing to become the Isochronous Resource manager will indicate this fact in their self ID packet. The highest numbered node that wishes to become resource manager will achieve the honor.

Normal Arbitration

Once the configuration process is complete, normal bus operations can begin. In order to fully understand arbitration a knowledge of the cycle structure of 1394 is necessary.

A 1394 cycle is a time slice with a nominal 125usec period. The 8 KHz cycle clock is kept by the cycle master, which is also the root node. To begin a cycle, the cycle master broadcasts a cycle start packet, which all other devices on the bus use to synchronize their timebases.

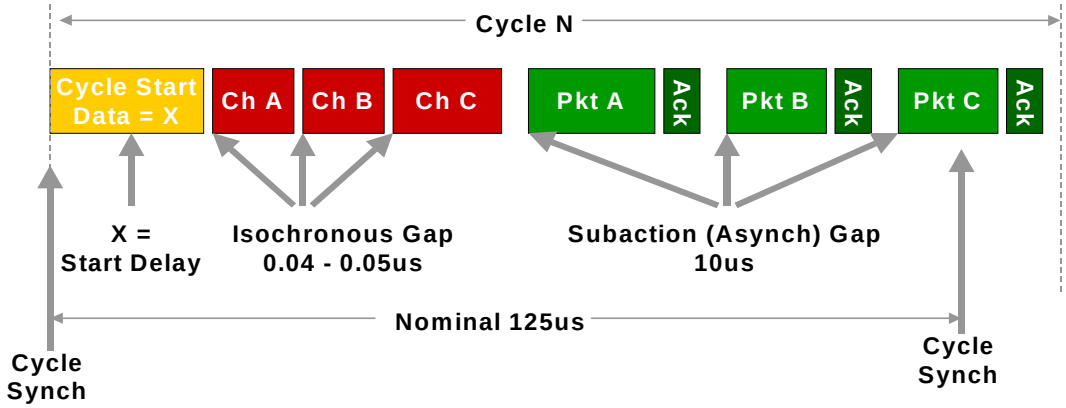

Figure 8 Typical 1394 Cycle

Questra CONSULTING

Immediately following the Cycle Start packet, devices that wish to broadcast their isochronous data may arbitrate for the bus. Arbitration consists of signaling your parent node that you wish to gain access to the bus. The parent nodes in turn signal their parents and so on until the request reaches the Root Node. In our example, suppose the Digital Camera and the PC wish to stream data over the bus. They both signal their parents that they wish to gain access to the bus. Since the PC's parent IS the Root Node, its request is received first and it is granted the bus. From this scenario, it is evident that the closest device to the Root Node wins the arbitration. Since Isochronous channels can only be used once per cycle, when the next Isochronous Gap occurs, the PC will no longer participate in the arbitration. This allows the Digital Camera to win the arbitration this time. Note that the PC could have more than one Isochronous channel, in which case it would win the arbitration until it had no more channels left. This points out the important role of the Isochronous resource manager – it will not allow the allotted isochronous channels to require more bandwidth then is available.

When the last isochronous channel has had transmitted its data, the bus goes idle waiting for another isochronous channel to begin arbitration. Since there are no more isochronous devices left waiting to transmit, the idle time extends longer than the isochronous gap until it reaches the duration defined as the subaction (or asynchronous) gap. It is at this time that asynchronous devices may begin to arbitrate for the bus. Arbitration proceeds in the same manner, with the closest device to the Root Node winning arbitration.

This brings up an interesting scenario – since asynchronous devices can send more than one packet per cycle, it might be possible for a device closest to the Root Node (or the Root Node itself) to hog the bus by always winning the arbitration. This scenario is dealt with using what is called the fairness interval and the Arbitration Rest gap. The concept is simple – once a node wins the asynchronous arbitration and delivers its packet, it clears its arbitration enable bit. When this bit is cleared, the physical layer no longer participates in the arbitration process, giving devices further away from the Root Node a fair shot at gaining access to the bus. When all devices wishing to gain access to the bus have had their fair shot, they all wind up having their arbitration enable bits cleared, meaning no one is trying to gain access to the bus. This causes the idle time on the bus to go longer than the 10us subaction gap until it finally reaches 20us, which is called the arbitration reset gap. Once the idle time reaches this point all devices may reset their arbitration enable bits and arbitration can begin all over again.

Link Layer

The Link Layer is the interface between the transaction layer and the physical layer. The Link Layer is responsible for checking received CRCs and calculating and appending the CRC to transmitted packets. In addition, since isochronous transfers do not use the transaction layer, the Link Layer is directly responsible for sending and receiving isochronous data. The Link Layer also examines the packet header information and determines the type of transaction that is in progress. This information is then passed up to the transaction layer. The interface between the Link Layer and the Physical Layer is listed as an informative (not required) Appendix in the 1395-1995 specification. In the 1394a addendum, this interface becomes a required part of the specification. This was done to promote interoperability amongst the various 1934 chip vendors.

The Link Layer to Physical Layer interface consists of a minimum of 17 signals that must be either magnetically or capcitively isolated from the PHY. These signals are defined in Table 2.

Questra CONSULTING

Signal Name	Source	Description
LReq	Link Layer	Link Request – used to initiate a request to send a packet as well as a request to read directly from a PHY register.
SClk	Physical Layer	49.152MHz clock used to synchronize data readout. (The frequency may change depending on data rates with 1394b)
Data[0:7]	Either	Data – higher transfer speeds use an increasing number of bits: 100Mbps – D[0:1] 200Mbps – D[0:3] 400Mbps – D[0:7] Note that the width of this data bus may exspand to 16 bits with 1394b.
Ctl[0:1]	Either	Control Interface – Defines what state the interface is in.
LPS	Link Layer	Link Power Status – Indicates that the Link Layer Controller is powered.
Link On	Physical Layer	Indicates that the Physical Layer has been powered on.
Direct	Neither	Indicates that no isolation barrier exists.
Backplane	Physical Layer	High if physical layer is a backplane implementation
Clk25	Neither	Indicates that SClk is only 24.576 MHz – valid in a backplane implementation only.

Table 2

A typical Link Layer implementation has the PHY interface, a CRC checking and generation mechanism, transmit and receive FIFOs, interrupt registers, a host interface and at least one DMA channel.

Transaction Layer

The transaction layer is used for asynchronous transactions. The 1394 protocol uses a request – response mechanism, with confirmations typically generated within each phase. There are several types of transactions allowed. They are listed as follows:

- Simple Quadlet (4 byte) Read
- Simple Quadlet Write
- Variable Length Read
- Variable Length Write
- Lock Transactions

Lock transactions allow for atomic swap and compare and swap operations to be performed.

Asynchronous packets have a standard header format along with an optional data block. The packets are assembled and dis-assembled by the Link Layer controller. Figure 9 shows the format of a typical asynchronous packet.

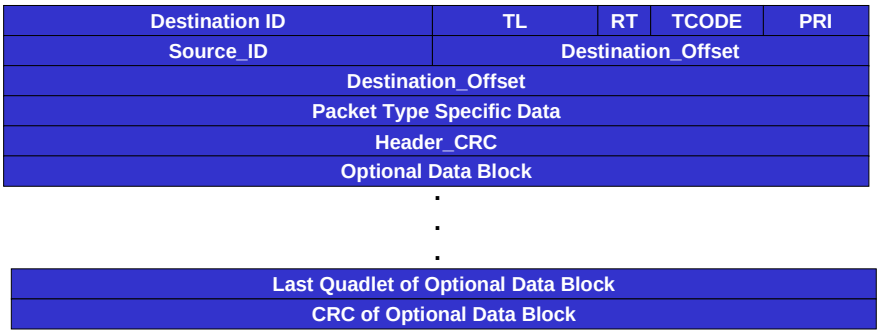

Figure 9 Asynchronous Packet Format

Name	Description
Destination_ID	The concatenation of the Bus and Node address of the intended node. All ones indicate a broadcast transmission.
TL	Transaction Label specified by the requesting node. This value is also used in the response packet.
RT	Retry Code that defines whether this is a retry and what retry mechanism is being used.
TCODE	Transaction Code defines the type of transaction (Read request, read response, Acknowledge, etc..)
PRI	Priority – used only in backplane environments
Source_ID	Specifies Bus and Node that generated this packet.
Destination_Offset	The address location within the destination node that is being accessed.
Packet type Specific Data	Can indicate data length for block reads and writes, or contain actual data for a quadlet write request or quadlet read response.
Header_CRC	CRC Value for the Data
Optional Data	Quadlet aligned data specific to the type of the packet.
Optional Data CRC	CRC for the Optional Data

Transactions can be split, concatenated or unified. A split transaction is shown in Figure 10. The split transaction occurs when a device cannot respond fast enough to the transaction request. When a request is received, the node responds with an acknowledge packet. An acknowledge packet is sent after every asynchronous packet. In fact the acknowledging device does not even have to arbitrate for the bus, control of the bus is automatic after receiving an incoming request or response packet.

Questra CONSULTING

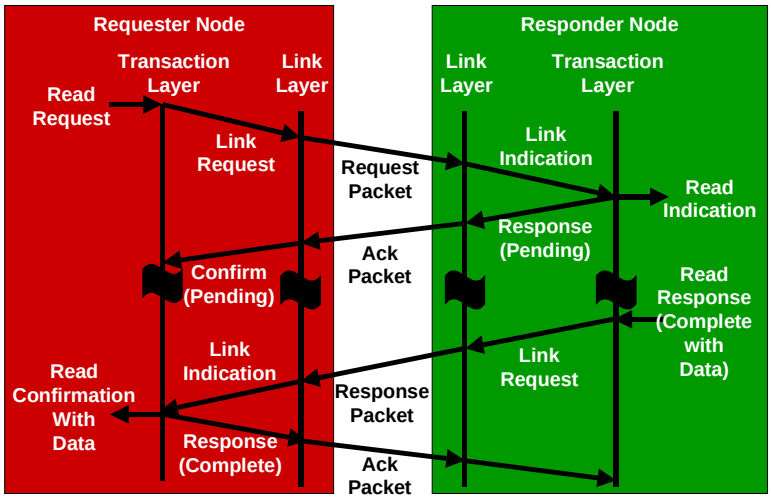

Figure 10 A Split Transaction

In Figure 10, the responder node sends the Acknowledge back and then prepares the data that was requested. While this is going on, other devices may be using the bus. Once the responder node has the data ready, it begins to arbitrate for the bus in order to send out its response packet containing the desired data. The requester node receives this data and returns an acknowledge packet (without needing to re-arbitrate for the bus).

If the responder node can prepare obtain the requested data fast enough, the entire transaction can be concatenated. This removes the need for the responding node to re-arbitrate for the bus after the acknowledge packet is sent.

For data writes it is also possible for the acknowledgement to be the response to the write. This is the case in a unified transaction. If the responder can accept the data fast enough, its acknowledge packet can have a transaction code of complete instead of pending. This eliminates the need for a separate response transaction altogether. Note that Unified Read and Lock transactions are not possible, the Acknowledge packet cannot return data. Figure 11 shows the different types of transactions supported by 1394.

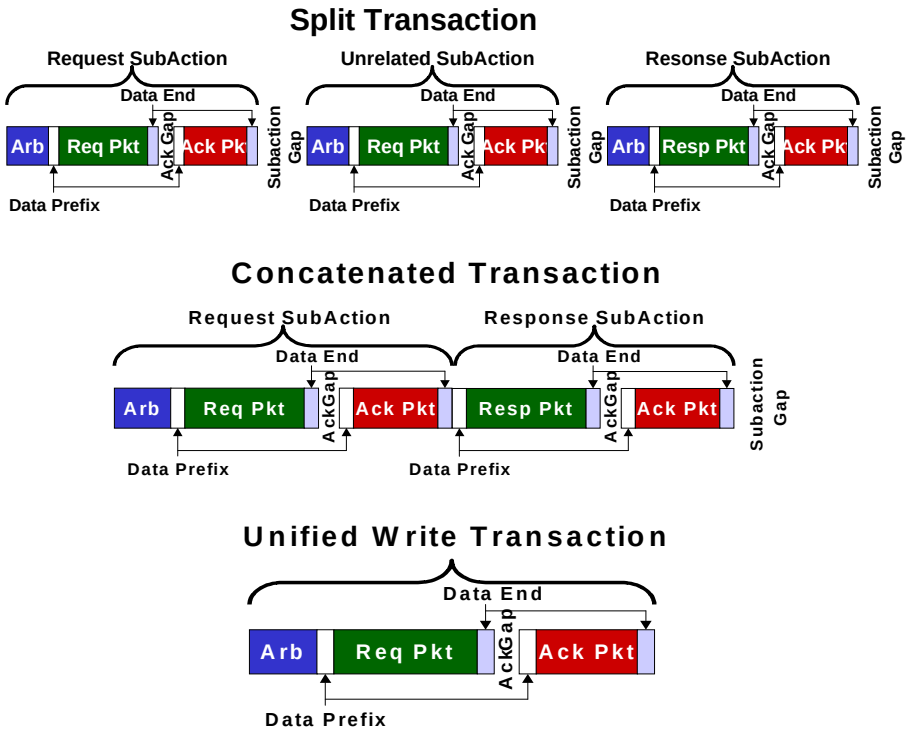

Figure 11 IEEE 1394 Transaction Types

1394a Arbitration Enhancements

The 1394a addendum adds three new types of arbitration to be used with asynchronous nodes.

Acknowledged Accelerated Arbitration – when a responding node also has a request packet to transmit, the responding node can immediately transmit its request without re-arbitrating for the bus. Normally the responding node would have to go through the normal arbitration process again.

Fly-by Arbitration – Nodes that contain several ports must act as a repeater on its active ports. A multiport node may use fly-by arbitration on packets that do not require acknowledgement (isochronous packets and acknowledge packets). When a node using this technique is repeating a packet UPSTREAM toward the Root Node, it may concatenate an identical speed packet to the end of the current packet. Note that asynchronous packets may not be added to isochronous packets.

Token-style Arbitration –token style arbitration requires a group of cooperating nodes. When the cooperating node closest to the Root Node wins a normal arbitration it can pass the arbitration grant down to the node furthest from the root. This node sends a normal packet, all of the cooperating nodes can use fly-by arbitration to add their packets to the original packet as it heads upstream.

Bus Management

Bus management on a 1394 bus involves several different responsibilities that may be distributed among more than one node. Nodes on the bus must assume the role of Cycle Master, Isochronous Resource Manager and Bus Manager.

Questra CONSULTING

Cycle Master

The cycle master initiates the 125us cycles. The Root Node MUST be the cycle master, if a node that is not cycle master capable becomes Root Node, the bus is reset and a node that is cycle master capable is forced to be the root. The cycle master broadcasts a cycle start packet every 125us. Note that it is possible for a cycle start to be delayed while an asynchronous packet is being transmitted or acknowledged. The cycle master deals with this by including the amount of time that the cycle was delayed in the cycle start packet.

Isochronous Resource Manager

The isochronous resource manager must be isochronous transaction capable. In addition the isochronous resource manager must also implement several optional registers. These registers include the Bus Manager ID register, the bus bandwidth allocation register and the channel allocation register. Isochronous channel allocation is performed by a node that wishes to transmit isochronous packets. These nodes must allocate a channel from the channel allocation register by reading the bits in the 64 bit register. Each channel has one bit associated with it – channels are available if the bit is set to a logic "1". The requesting node sets the first channel bit available to a logic "0" and uses this bit number as the channel ID.

In addition, the requesting node must examine the Bandwidth Available Register to determine how much bandwidth it can consume. The total amount of bandwidth available is 6144 allocation units. One allocation unit is the time required to transfer one quadlet at 1600Mbps. There is a total of 4915 allocation units available for isochronous transfers if any asynchronous transfers are used. Nodes wishing to use isochronous bandwidth must subtract the amount of bandwidth needed from the Bandwidth Available Register.

Bus Manager

A bus manager has several functions including publishing a topology map and a speed map, power management and optimizing bus traffic. The speed map is used by nodes to determine what speed it can use to communicate with other nodes. The topology map may be used by nodes with a sophisticated user interface that could instruct the end user on the optimum connection topology to enable the highest throughput between nodes. The bus manager is also responsible for determining whether the node that has become Root Node is cycle master capable. If it is not, the bus manager searches for a node that is cycle master capable and forces a bus reset that will select that node as Root Node. There may not always be a bus manager capable node on a bus, in this case at least some of the bus management functions are performed by the isochronous resource manager.

Hardware and Software Support

Hardware

Several manufacturers make devices for engineers designing devices that support IEEE 1394. Integrated circuit providers typically provide a chipset that includes a Link Layer controller and a Physical Layer controller. One of the goals of the 1394a addendum is to provide interoperability among the various Link Layer and Physical Layer controllers. Some of the available ICs and cores are shown in Table 3.

Link Layer Controllers		
Manufacturer	Part Number	Description
Fujitsu Microelectronics	MB8661x	Combined Link/PHY Core & ICs
IBM	IBM21S650PFA	PCI Based Link Layer Controller

Questra CONSULTING

	IBM21S550PFB	Generic Bus Interface Link Layer Controller
Innovative Semiconductor	SL75x	Link Layer Cores
Philips Semicondutor	PDI1394L11	A/V Link Layer Controller
Phoenix Technologies	VirtualLink	1394a Compatible Link Layer Cores
Sand	1394 Device Controller	1394 Link Layer Core
Symbios	SYM13FW600	PCI Bus Interface Link Layer
	SYM13FW500	1394 to ATA/ATAPI Interface
Texas Instruments	TSB12LV21B TSB12LV22 TSB12LV31	LynxHCI (PCI) IC OHCI (PCI) IC General Purpose Bus Interface IC
Physical Layer Controllers		
Manufacturer	Part Number	Description
Fujitsu Microelectronics	MB8661x	Combined Link/PHY Core & ICs
IBM	IBM21S85xPFD	400Mbps 1 and 3 port devices
	IBM21S760PFD	200Mbps 1 and 3 port devices
Innovative Semiconductor	SL75x	Physical Layer Cores
Philips Semiconductor	PDI1394P11	Physical Layer IC
MacroDesigns	-	Physical Layer Cores
Phoenix Technologies	VirtualLink	100,200 &400Mbps 1394a Compatible Cores
Sand	1394 CPHY	1394 Cable Physical Layer Core
Symbios	SYM13FW403	1394 Cable PHY Interface IC
Texas Instruments	TSB11C01 TSB11LV01 TSB14C01A TSB21LV03A TSB41lV0x	Up to 400Mbps PHY ICs

Table 3

As well, complete PCI based cards that plug in to a PC backplane are available from companies such as Adaptec, Sony and Texas Instruments.

Software Support

IEEE 1394 is directly supported in the new Windows Driver Model (WDM) which is used in Windows 98 and will be available in Windows NT 5.0. In order for chipsets and devices to support the drivers provided in the new versions of Windows, several members of the 1394 Trade Association have banded together to create the 1394 Open Host Controller Interface (OHCI) Specification. The OHCI provides a Link Layer controller as well as Bus Management functionality. In addition, the OHCI defines several DMA controllers for Asynchronous and Isochronous transactions. These controllers provide registers that a standard 1394 driver provided by Microsoft can use to configure the controller and schedule transactions.

Microsoft provides WDM Streaming drivers for supporting Audio and Video devices such as DVD players, MPEG decoders, tuners and audio codecs. These streaming drivers allow low latency support for isochronous channels. The drivers minimize transitions between user mode and kernel mode which significantly reduces the overhead for driver calls and data movement.

For Storage Devices, Printers and Scanners, Windows NT 5.0 supports the Serial Block Protocol (SBP-2). Microsoft recommends that devices be written to support the SCSI command set so the device can use the existing

Questra CONSULTING

SCSI class driver that sits on top of the SBP-2 driver. If the vendor does not support the SCSI protocol, they will need to write their own Class driver to support their own command set.

In addition to the SBP-2 specification for storage devices, there are other standard data formats that ride on top of 1394 that are in various stages of completion. These include the Tailgate specification that defines a method for adapting ATA/ATAPI controllers to 1394, a Digital Video (DV) standard, the Digital Still Image working group, a Printer Protocol and an Industrial Control and Instrumentation group.

Embedded systems designers have also seen some RTOS vendors claim support for 1394 including Integrated Systems pSOS and Wind River's VxWorks among others. These OS vendors typically support a third party protocol stack that has been ported to their OS. In the case of 1394, Zayante, Award Software and Technology Rendevous each have a 1394 stack that they claim is OS independent. Windows CE does not currently have native support 1394 but it will undoubtedly support it in the very near future. There is third party support to fill the existing gap.

Conclusion

The IEEE 1394 protocol, along with USB, Ethernet and IrDA will be the data channels of the future. Any embedded system that needs to share information (and what applications won't?) will use at least one of the aforementioned communication methods. IEEE 1394 provides the highest throughput as well as providing isochronous capability and peer to peer support. These features make it a prime candidate as the driver for the consumer, computer and communications convergence. Proposed enhancements and additions to the protocol are targeting higher speeds, home networking, fiber transmission wireless IR transmission. As more devices look to support 1394, the prices for silicon support are dropping rapidly, which will in turn cause more engineers to design in the protocol.

References & Useful Sites

1. "FireWire System Architecture" – Don Anderson, MindShare Inc., Addison-Wesley, ISBN 0-201-69470-0
2. IEEE 1394-1995 Serial Bus Specification
3. ISO/IEC 13213 (ANSI / IEEE 1212) CSR Architecture Specification
4. www.1394ta.org – The 1394 Trade Association Home Page
5. www.microsoft.com/hwdev/busbios/1394support.htm – Microsoft's 1394 Support
6. ftp://ftp.austin.ibm.com/pub/chrptech/1394ohci – The 1394 OHCI Specification
7. http://www.ti.com/sc/docs/msp/1394/1394.htm – TI's 1394 Home Page