

Serial ATA Design and Validation Challenges

by Marco Davila, Agilent Technologies

As serial ATA (SATA) moves from 1.5 Gbit/s to 3.0 Gbit/s, and with serial attached SCSI (SAS) starting at 3 Gbit/s, designers are discovering inherent difficulties in trying to design a reliable link (ie jitter caused by issues like inter-symbol interference (ISI), crosstalk, cable losses, transmission line effects, etc). In addition, trying to reliably probe the link, with full visibility into things like out-of-band signaling and the ability to work with spread-spectrum clocking, can be very difficult. Let's examine these issues more closely and look at the common tools that help alleviate these challenges.

Channel Simulation

One of the most important steps in creating a reliable SATA/SAS link starts at its most basic level: the physical layer. As speeds approach RF, careful design and planning become more and more crucial. Hardware designers must use tools that can model their signal path (channel) effects. These effects directly impact the performance and quality of the serial link. Lossy transmission lines and cable losses can be compensated for by the use of pre-emphasis, but other effects such as impedance discontinuities, ISI, and crosstalk need closer attention. Advanced circuit simulation techniques can be used to uncover these issues before they become potential problems.

Tools such as Agilent EEsof EDA's Advanced Design System can model vias, lossy transmission lines, and cable interconnects. With these tools hardware engineers can get a very accurate representation of these effects and plan accordingly to prevent eye closures at their respective receivers. Advanced PCB design techniques can be used to combat these signal integrity (SI) issues. At RF vias look like stubs and PCB manufacturers can back-drill them to remove the majority of the stub via barreling techniques (see Fig. 1).

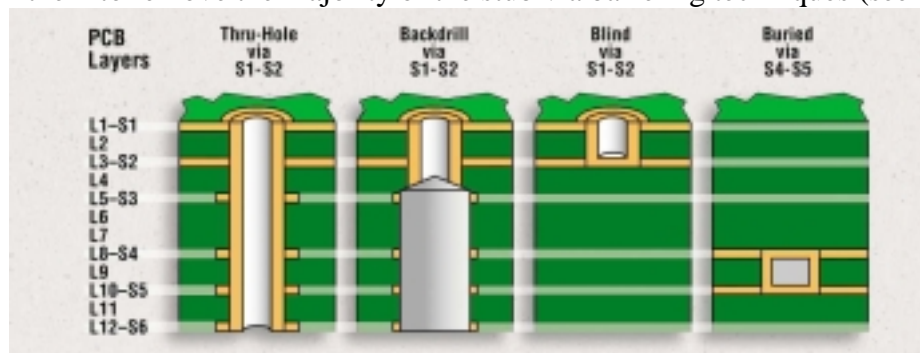


Fig. 1: Example Of Various Via Types

Also, other techniques can be used, such as buried or blind via constructions. In addition inductive connectors can be compensated by using wider pads which add capacitance to closely match the characteristic impedance of the channel. Reducing these discontinuities will greatly improve SI effects in the channel and lead to increased margin within the link.

Physical-Layer Testing

During board turn-on/characterization, physical-layer testing becomes an important part of the design process. Measurements must be made to verify models and design goals. One of the most important measurements is jitter. Measuring the amount of eye height and width on the receiver input are figures needed to determine how much margin the receiver has. When determining which oscilloscope to use, one with clock recovery and mask unfolding capabilities to recreate the transmitted eye will lead to an accurate jitter measurement. Making an edge measurement as shown in Fig. 2 just will not do as it will give you a false indication of jitter. Typically you want your oscilloscope bandwidth to be 4x the fundamental frequency of the signal you are measuring. For example, serial ATA gen2 at 3.0 Gbit/s gives a fundamental frequency of 1.5 GHz, so 6 GHz is the minimum recommended bandwidth for this type of measurement.

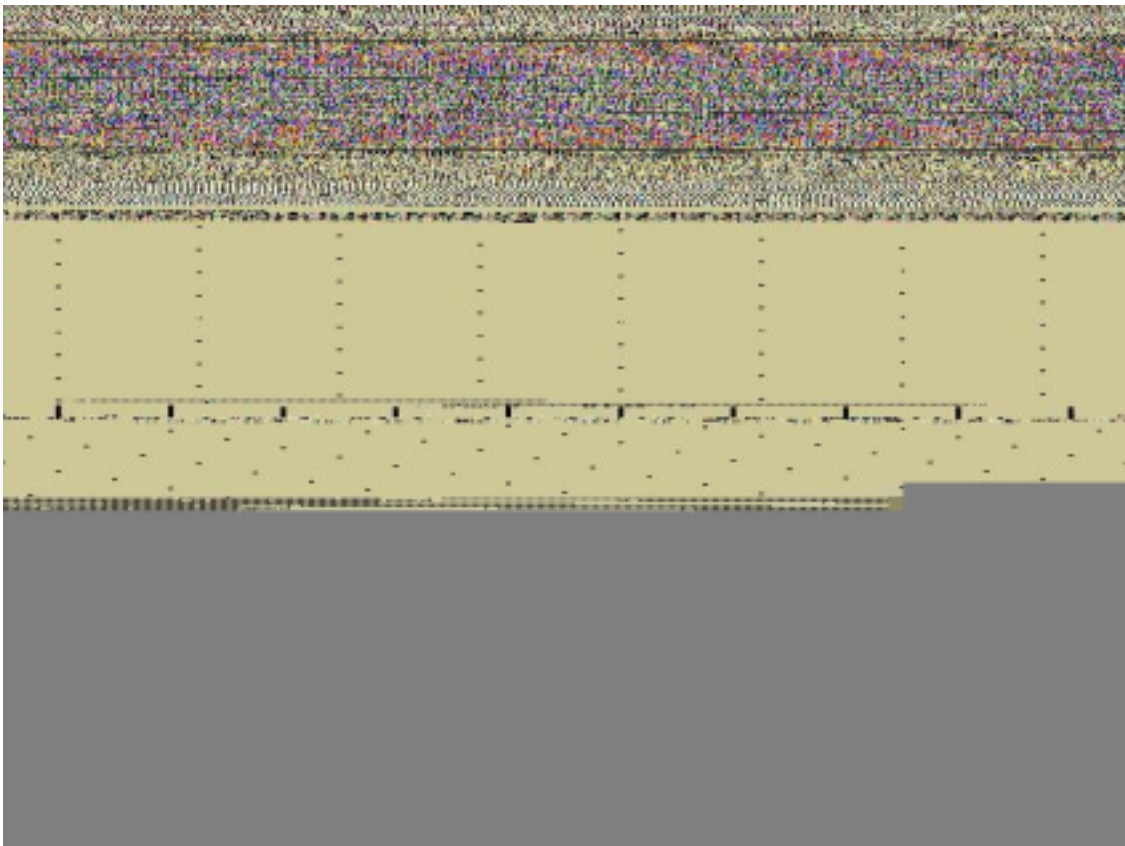


Fig. 2: Edge Measurement

When transmitting over long distances, there is a notion of pre-emphasis and de-emphasis, which is used by the transmitter to counter-act the effects of a lossy channel. As shown in Fig. 3 these pre-emphasis/de-emphasis bits should be measured with an oscilloscope to determine if too much is being used. Yes, too much pre-emphasis/de-emphasis can also cause eye closures. If the channel is not very lossy, higher pre-emphasis levels are not absorbed by the channel, which causes ISI and other transmission line phenomena.

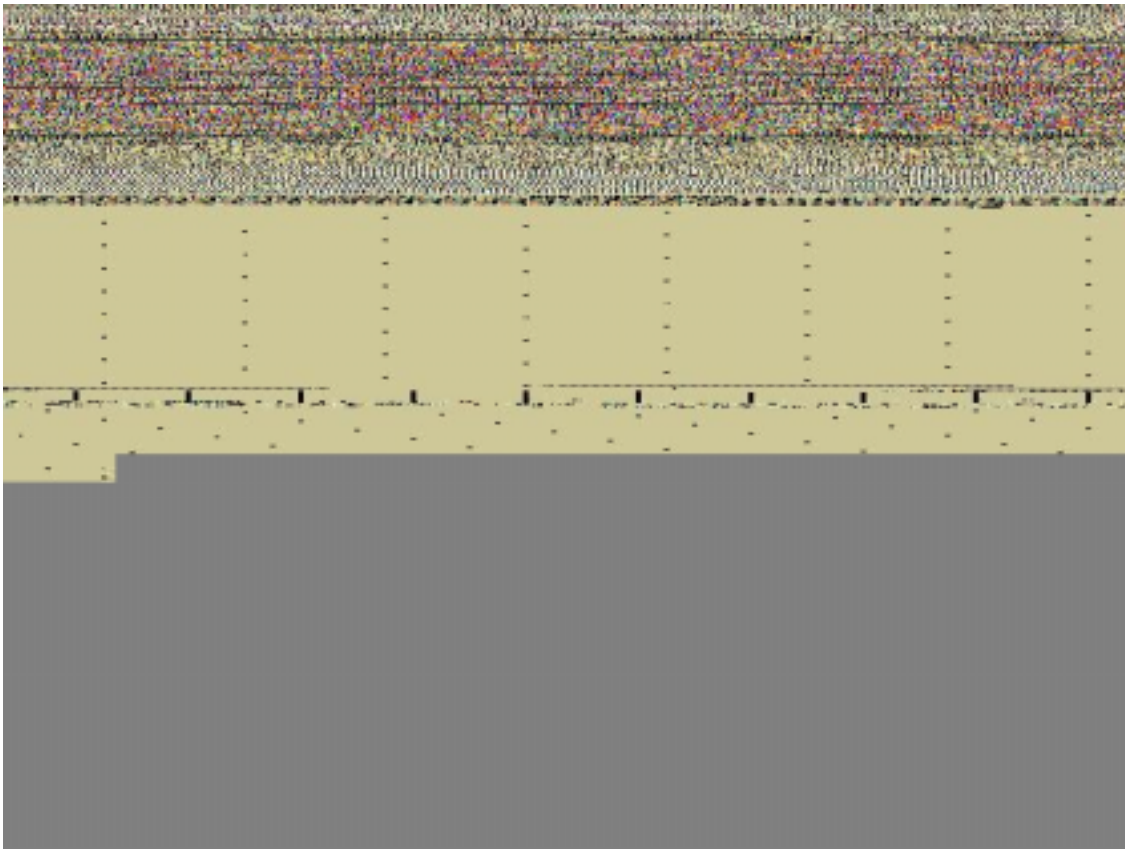


Figure 1. A serial ATA 3.0-Gbit/s data eye with pre-emphasis

Connecting to the link is also important. The best place to probe is at the receiver input terminations -- or the transmitter output terminations -- with a high-bandwidth ultra-low load probe. But these measurements can also be made with break-out boards or using an ultra-low load interposer.

Compliance test tools can be a great help to determine if your host/device is serial ATA compliant. These compliance test tools come with software packages available with some oscilloscopes (Fig. 4). The tools simplify the testing and give a pass/fail report.

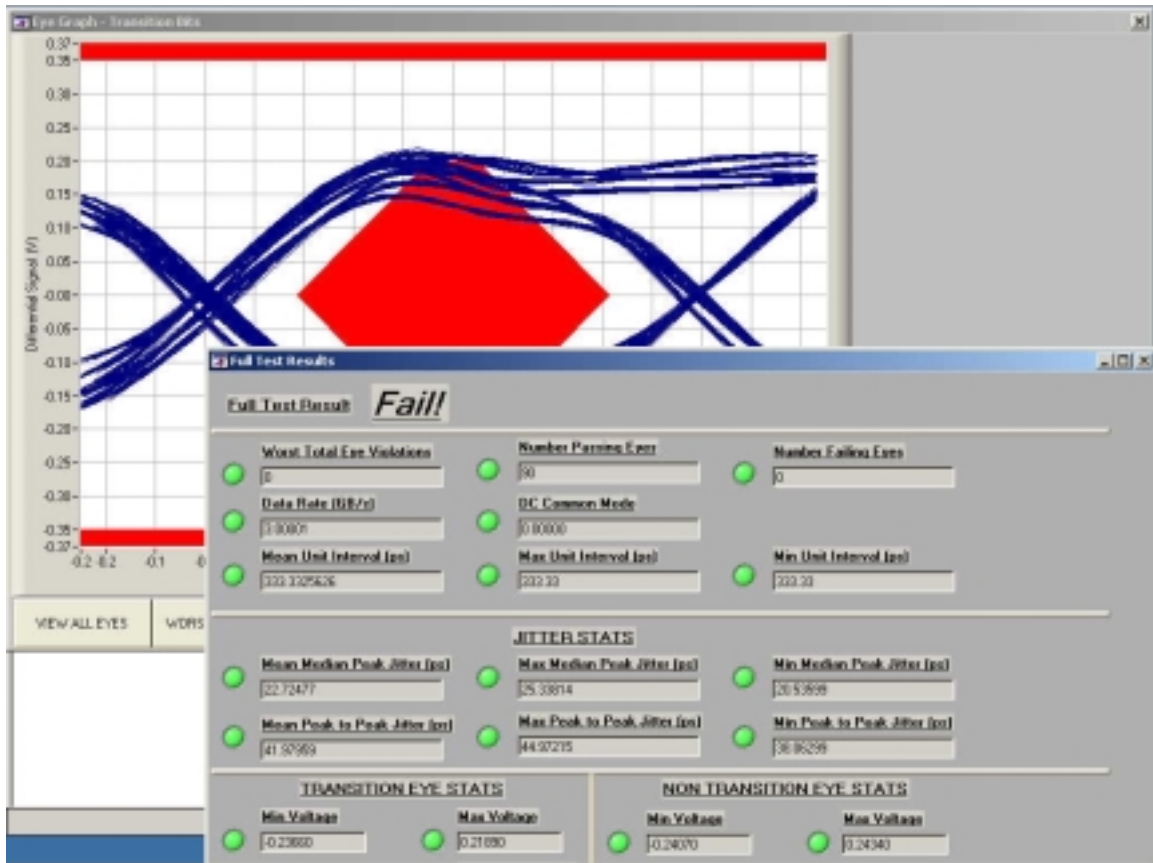


Fig. 4: Automated Compliance-Test Software Package

SATA has a method of link negotiation by means of out-of-band (OOB) signaling. Equipment is available on the market that can test a device or host for correct OOB signaling behavior (Fig. 5). These automated test tools can test the specified squelch window to determine whether these OOB signals are COMINIT, COMWAKE, or others (Fig. 6). These are great tools in determining how well the device under test (DUT) adheres to the SATA specifications.

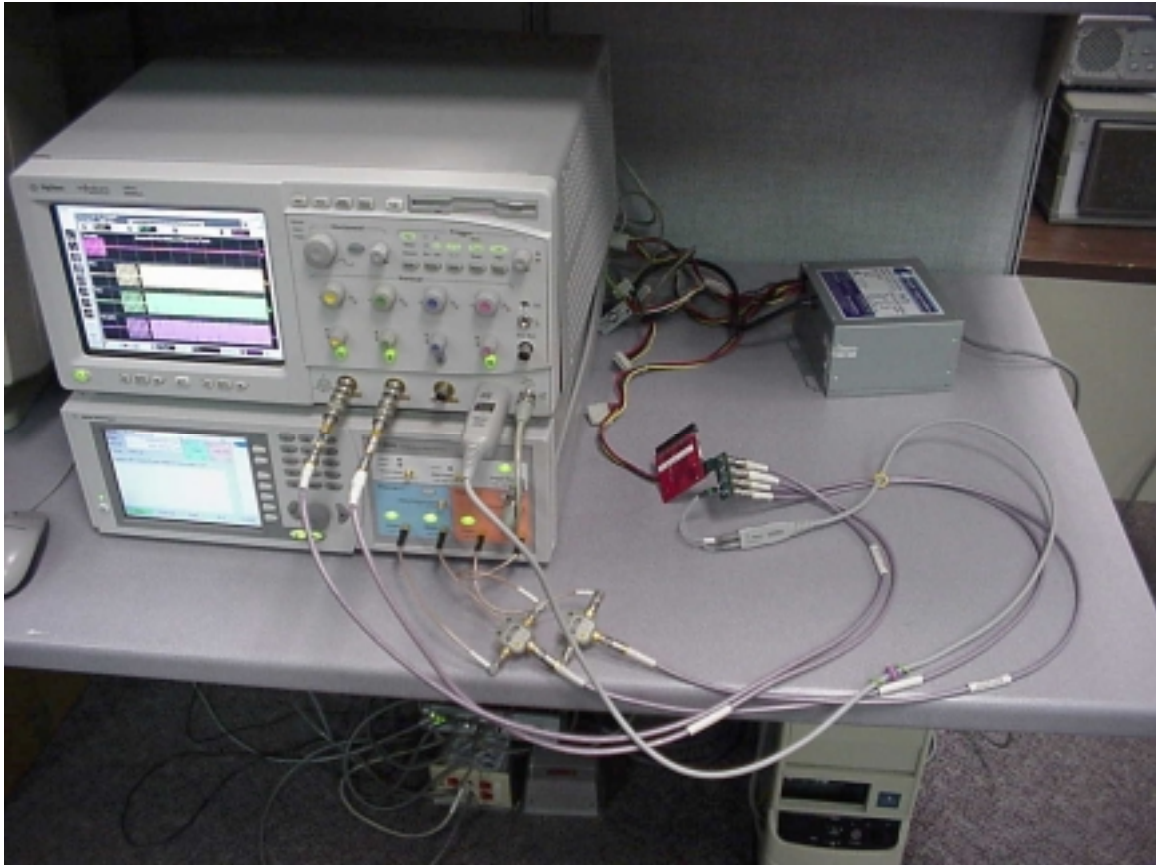


Fig. 5: Automated Test Using Pattern Generator And Oscilloscope

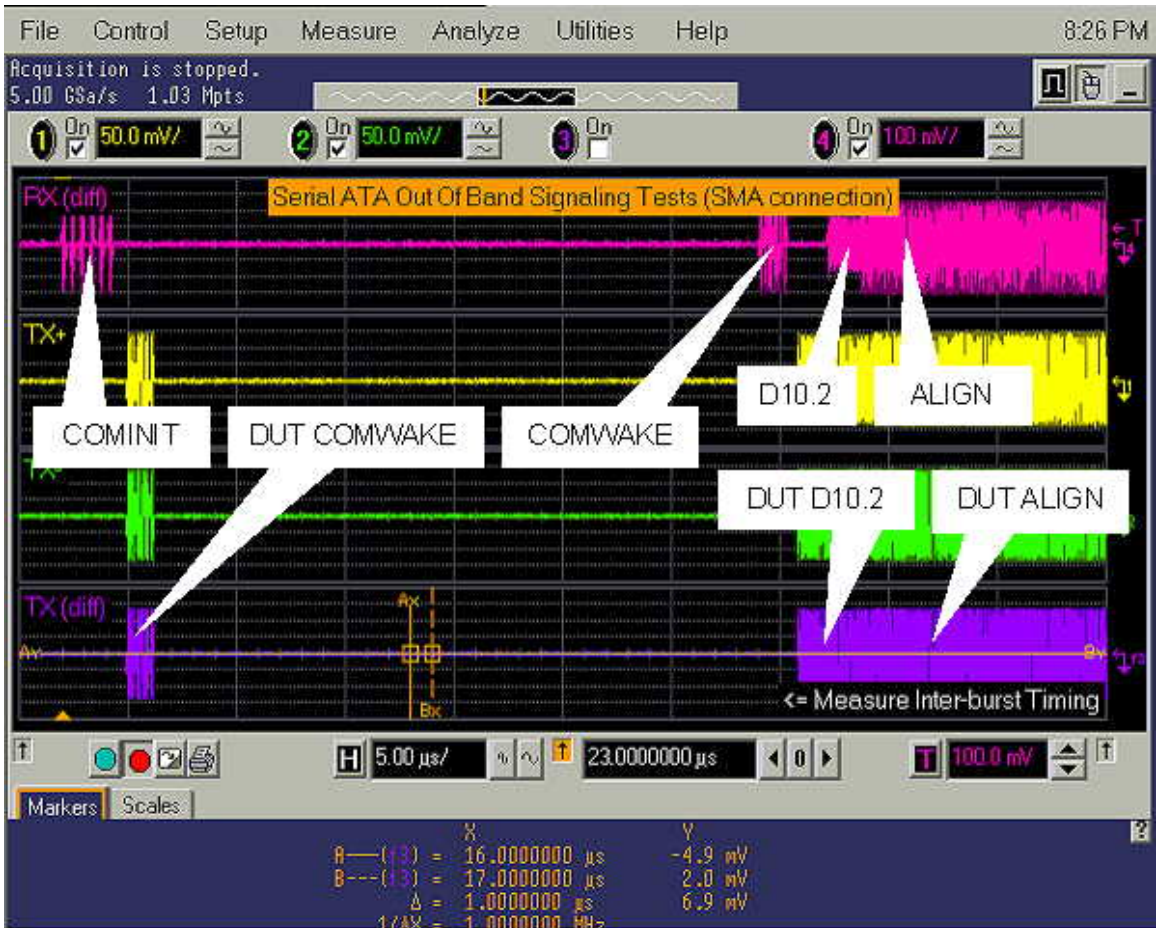


Fig. 6: Compliance Test Results In OOB Capture

Protocol-Layer Testing

Analyzer topologies

The requirement here is a higher-level view of a SATA link. These tools give visibility into primitives and packets occurring across a link. There are many test tools available for protocol testing. The following are example topologies of current analyzer designs and the advantages/disadvantages of each.

A repeating analyzer topology design re-times and repeats the signal on to the next device (Fig. 7). This design physically breaks the link and act as an endpoint. The analyzer must be electrically SATA-compliant.

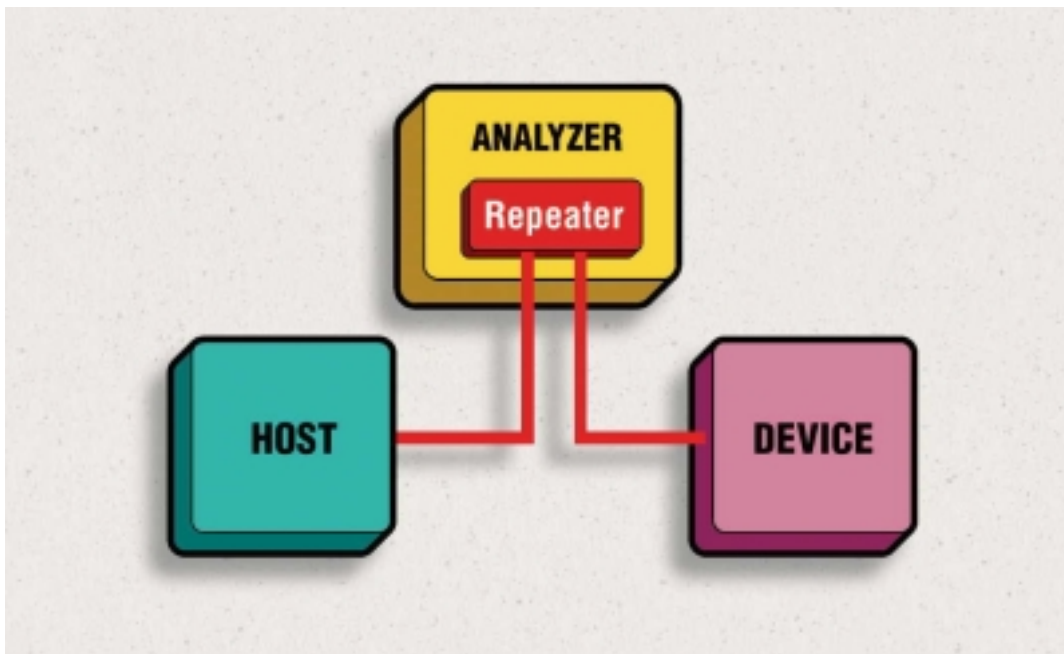


Fig. 7: Repeating Analyzer Topology

The advantages of such an approach is the elimination for the need of a high-impedance probe, since high-impedance probes are more difficult to design. A repeater also does not require visibility into the middle of the link. The full signal is captured and terminated in the characteristic impedance of the link, then repeated.

The disadvantages of this design are that real-life issues may be masked (eg works with the repeater, not without it). The analyzer now develops the transmitted signal, not the DUT. This has a direct impact on jitter since the analyzer is transmitting “repeating” the captured signal. Characteristics of the analyzer are now more critical; not only are you measuring the system jitter but also that of the analyzer. This is an important factor in overall system performance. The timing of the system is also affected since a re-timing repeater “re-times” the signal with respect to its own timebase. This re-timing is done by a CDR which itself has lock time, spread-spectrum clocking (SSC), and jitter tolerance specifications which must match that of the SATA/SAS specification. Another aspect of

this design is that the analyzer must be OOB transparent so the host and device can operate properly.

A snoop analyzer topology design taps the signal with a high-impedance probe (Fig. 8). The analyzer is not an endpoint. This design “snoops” the traffic occurring across a link.

The advantages of such an approach are the non-intrusive nature of this design. This analyzer is desired since it does not affect the link, timing remains intact, and it is OOB transparent. The analyzer is by design SATA/SAS-compliant since the link is never broken. The rule in analyzer design is first and foremost not to break the DUT, then second is to make the measurement reliably and accurately.

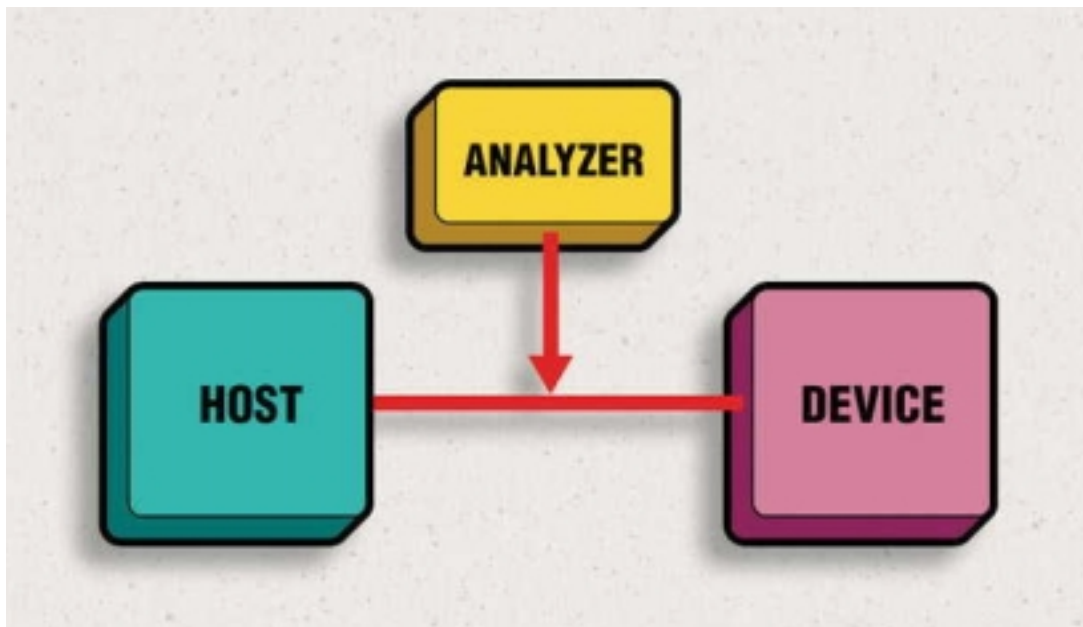


Fig. 8: Snooping Analyzer Topology

The disadvantages of this design are that it requires a high-impedance, ultra-low loading probe, which is more difficult to design. Also the requirement here is for visibility into middle of link.

From these design topologies one can see that a snoop design is the preferred method to see if a data eye exists in the middle of the link. Because a snooping design is capturing both transmit and receive traffic, the ideal place to capture the traffic would be at the end points like you would do with an oscilloscope measurement (ie receiver inputs and transmitter outputs). However, this does not lend itself to an interposer solution which requires capturing information in the middle of the link. An interposer solution requires capturing information in the middle of the link because it is a single tap point for both transmit and receive traffic. This can cause a problem, because the signal is collapsed in the middle of the link due to impedance mismatches, near and far end reflections, and ISI. This makes it difficult to design a probe to work in these channel environments.

However, for SATA gen2 3 Gbit/s and SAS gen1 3 Gbit/s this is not a problem as visibility into the middle of the link still remains viable. As speeds approach 6 Gbit/s, and beyond, this visibility will collapse due to these transmission line phenomena and other analyzer topologies will have to be considered.

Analysis probes

How does a 200-MHz logic analyzer capture serial ATA traffic at 3 Gbit/s? An analysis probe solves this move from parallel to serial for the logic analyzer. How does an analysis probe work (Fig. 9)? The serial data is captured at full speed; this stream of data is bit synchronized using a CDR type device, framed on a 10-bit boundary, and 8b/10b decoded. Then it is de-multiplexed to a logic analyzer.



Fig. 9: SATA gen2 Protocol Trace Capture Using Analysis Probe

Another aspect of the analysis probe is the connection scheme. The probing scheme must be mechanically and electrically non-intrusive. Fig. 10 illustrates a few probes available on the market today. The top probe requires the use of lengthy cables to egress out of the target. The bottom probe is an example of a small form-factor non-intrusive probe. Avoid connection schemes which require longer length cables as this is a violation of the specification. Instead of debugging your protocol/transaction layer you might introduce physical layer SI problems.

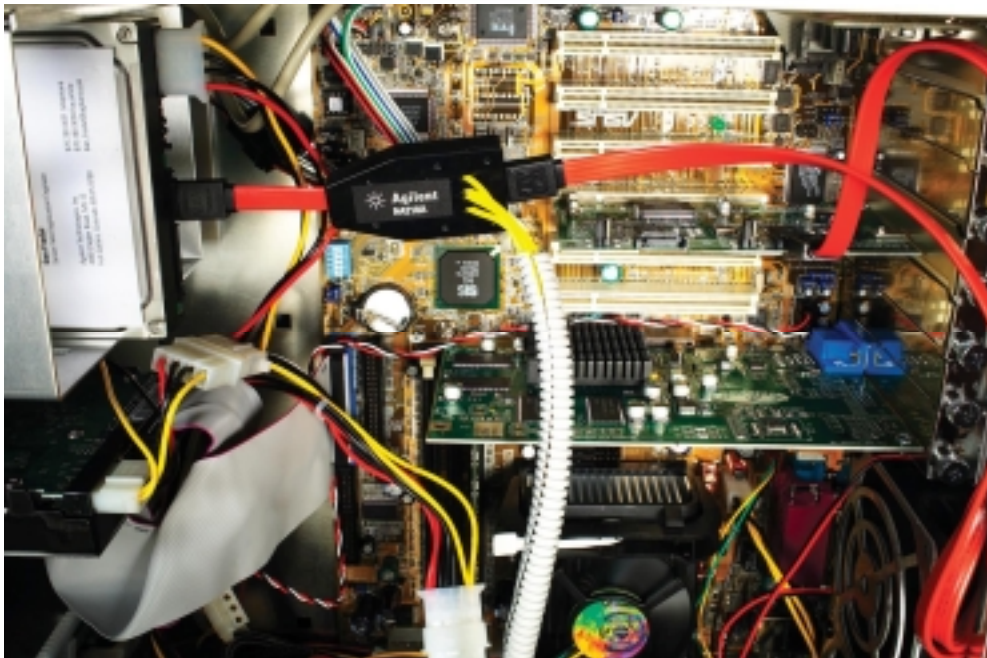


Fig. 10: Example Analyzer Probe Connections

Triggering advancements

Serialized links are difficult for a traditional logic analyzer to trigger on. In the parallel world data is captured wide and slow, the analyzer could then use its sequence levels to trigger on a particular ATA event. The sequence levels are finite and would be exhausted very quickly if one were trying to trigger on a packet of a serial link. Often, with system debugging, the need to trigger on specific packets is crucial to solving a problem. To solve the triggering issues associated with serial links, packet recognizers are used to trigger on specific packets. Having a means to trigger closer to the problem or perhaps the offending packet saves both time and money. With pinpoint triggering memory need not be as deep, which saves time since you wouldn't have to analyze very much data. Another useful trigger would be the ability to trigger on primitives themselves.

How does a packet recognizer work? Although packet recognizers are extremely powerful, they are actually very straightforward in the way they work. At first the packet recognizer must determine the start of the packet. The primitive defined for a SATA/SAS start of packet is automatically done without the engineer having to worry about defining the trigger steps to recognize this. Traditional logic analyzer triggering ends up using a large portion of its resources to determine only this event.

After resolving the start of a packet (just as the actual receiver does), the packet recognizes it then looks for matches to fields within the packet header and the data payload. The packet analysis probe will then send a signal back to the logic analyzer, which it can use as a trigger. These signals can be used with the full triggering resources of the analyzer (including counters, timers, sequences, storing, and multi-way branching) to provide very robust, powerful triggering. Fig. 11 shows a block diagram of the traditional approach to triggering using a logic analyzer, while Fig. 12 shows a block diagram of a logic analysis system using packet recognizers for triggering.

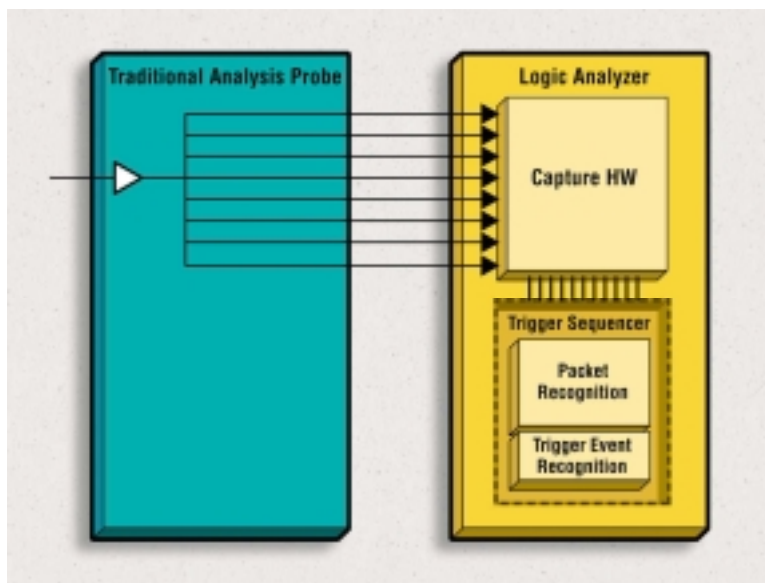


Fig. 11: Block Diagram Of Traditional Triggering Using Logic Analyzer

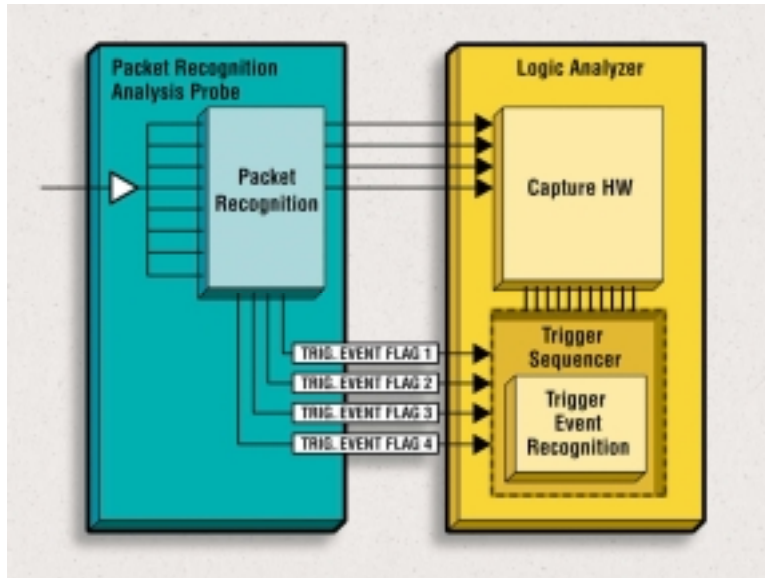


Fig. 12: Block Diagram Of Logic Analysis Using Packet Analysis Probe For Trigger

Common debug triggers

Using packet recognizers allows designers to define an almost limitless amount of triggers. They are often used in debug techniques like:

- Pre-store and qualified capture of packets
- Cross bus triggering
- Triggering using an exerciser

Each of these techniques will help an engineer as they bring up their new design.

Pre-store and qualified capture of packets

During the initial bring up of a SATA/SAS device or host, engineers will often want to capture a specific event and a large period of time before that event. Because of the need to capture a long period in time, it is often beneficial to only store events that are of interest in the logic analyzer's memory. However, this requires additional triggering and storage resources. If these resources are completely used defining the type of packet, this may not be possible.

Using a packet recognizer helps alleviate this problem (Fig. 13). For example, an engineer can define a specific packet header, along with several bytes of data. We will call this *data-bidirectional*. The engineer can then define another packet that includes all the types of events he wants to store. In this case we only want to store DMA Activate and Setup -- all other fields in the recognizer are left as "don't cares." We will call this "DMA only."

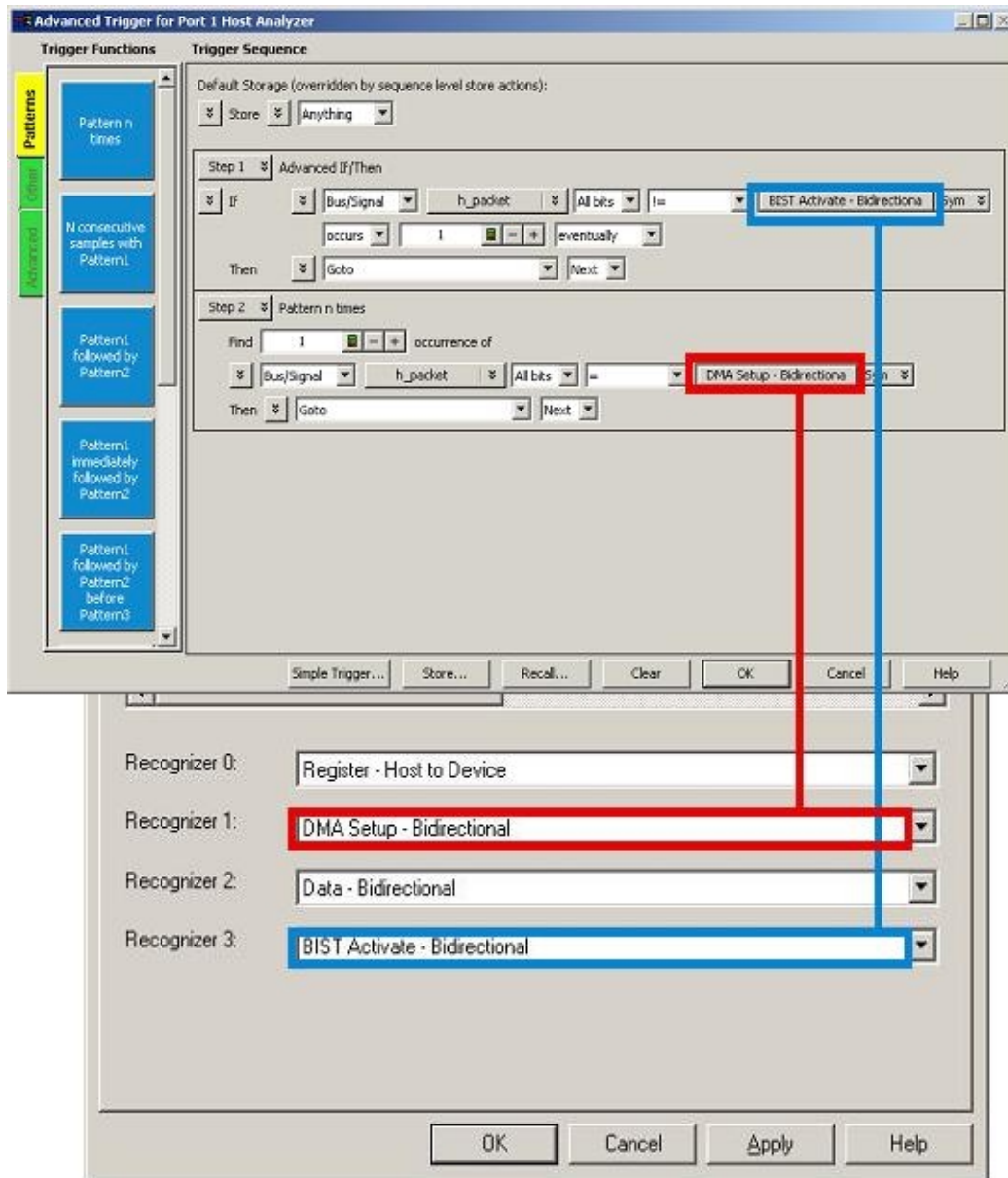


Fig. 13: Logic Analyzer Trigger Sequence

The logic analyzer will then use a simple pattern trigger to find the data-bidirectional event, and the engineer has all of the analyzer's resources left to qualify what is stored.

In this case the trigger will look like:

1. If #packet = "Data-Bidirectional" then
Trigger and fill memory

The analyzers default storing would look like:

Store #packet = "DMA only"

Often the designer will only want to see information before the trigger. In this case, the engineer can set the logic analyzer to do what is called *pre-store*. A 100% pre-store will only store information before the trigger, so you can capture a larger period of time before your trigger event. When used in conjunction with the default storing, this allows the engineer to capture the maximum amount of time before the trigger. In most logic analyzers the percent *pre-* or *post-store* is easily defined by the user.

Cross-bus triggering

In a serial architecture like SATA/SAS, a disagreement between the perceived traffic viewed by the transmitter and receiver doesn't always point to the root cause of a problem. Using a cross-bus triggering technique allows the engineer to not only trigger on this disagreement, but to also locate the source of problem. This problem might be caused by another bus in the system such as the processor system bus, DDR memory bus, PCI-Express bus or an I/O bus.

This is a very easy trigger to setup, but is very powerful in the information it provides. Designers can trigger from any one bus, and capture time-correlated events on the other buses in their system. For example, a common trigger involves looking for a bus-hang on the processor system bus. This will then trigger and capture data on all of the additional buses the designer is looking at. Should the processor bus-hang be caused by an event on the SATA/SAS link, this is a quick way to see the events all time correlated together for maximum debug.

Another common cross-bus triggering technique involves looking at a PCI-Express link from the North to the South bridge in which there lies a switch with SATA/SAS links. For example, it is often beneficial to trace a specific event as it occurs on the PCI-Express bus and travels through the bridge to the SATA/SAS link. Once again packet recognizers can be very beneficial in this case since they allow the designer to look for a very specific packet header with data. Traditional triggering using the logic analyzer's resources would have a difficult time defining the packet with enough detail to capture this event easily.

A Valuable Lesson

From the aforementioned discussion, we have learned that logic analyzers are not always the first piece of test equipment to come to mind when test equipment is needed for debugging and validating a serial link. However, special hardware triggering resources have made the logic analyzer a powerful tool in validating these designs. From the initial bring up of a device using a pre-store or *qualified capture of packets technique*, to a *stimulus and response* technique for compliance testing and final validation, the logic analyzer has come a long way in its 31 years.

For more information on the SATA/SAS packet analysis probe see <http://www.agilent.com/find/sata> Protocol tools: N4219B

About The Author

Marco Davila graduated from the University of Oklahoma with a BS degree in Computer Engineering with minors in Computer Science & Mathematics. He is presently pursuing an MS degree in Electrical Engineering at the University of Colorado. Marco started with Agilent Technologies (formerly Hewlett-Packard) in 1988 and is presently an R&D hardware design engineer. After work, his interests include snowboarding, mountain biking, hiking and golf.

