Testing SerDes I/O links the third time around

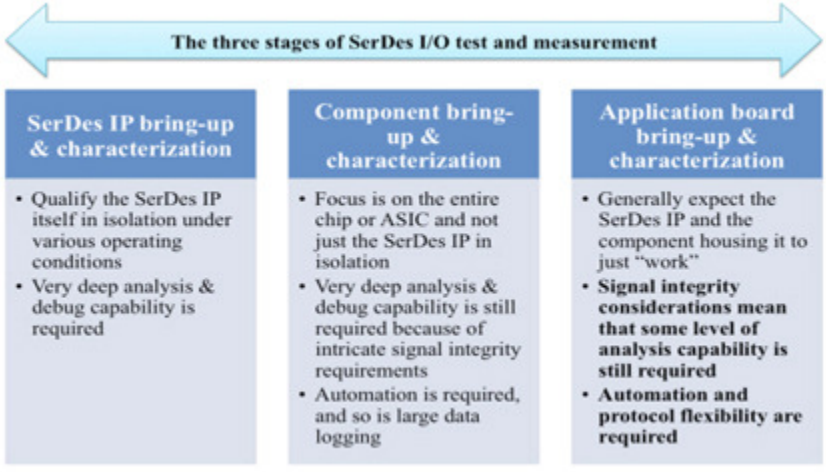
Dr. Mohamed Hafed, DFT Microsystems

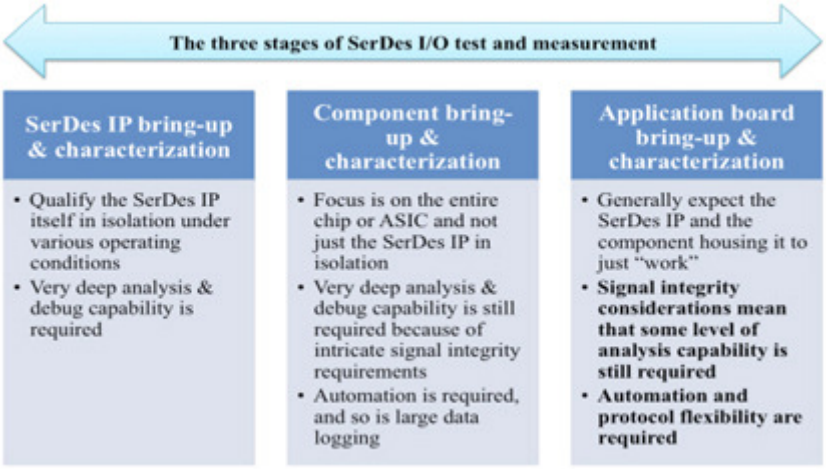
10/27/2010 10:57 AM EDT

From the title of this article, you probably assumed that it covers third-generation SerDes I/O interfaces such as PCI Express 3.0 or USB 3.0.

While the principles described here certainly apply to all protocol generations, this article is quite literally about testing a single SerDes IP for the third time in its existence! By the "third time," of course, we mean the so-called third instantiation of the SerDes IP, which corresponds to the assembly of a final SerDes-based semiconductor component on a system board or application board - see figure 1.

What kind of SerDes testing do we need at this final implementation stage? Are there specific goals, challenges, and tools that are needed at this point of integration? This article addresses these questions. We first describe the goals of test at the system level when compared to the component or IP levels. Then, we describe major tests and guidelines.





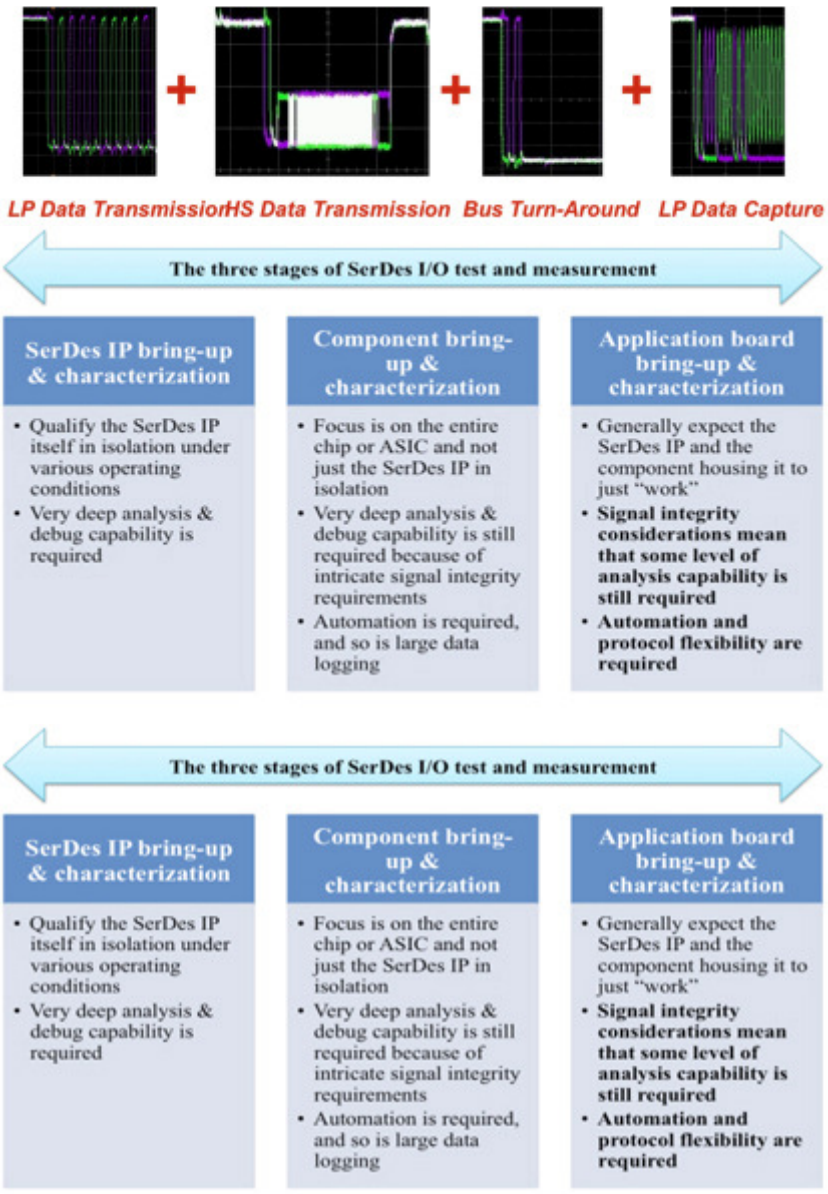


Fig 1: The three general stages during which a SerDes IP macro is measured or validated. The bottom panels show the general test requirements for each stage.

The third time is the charm

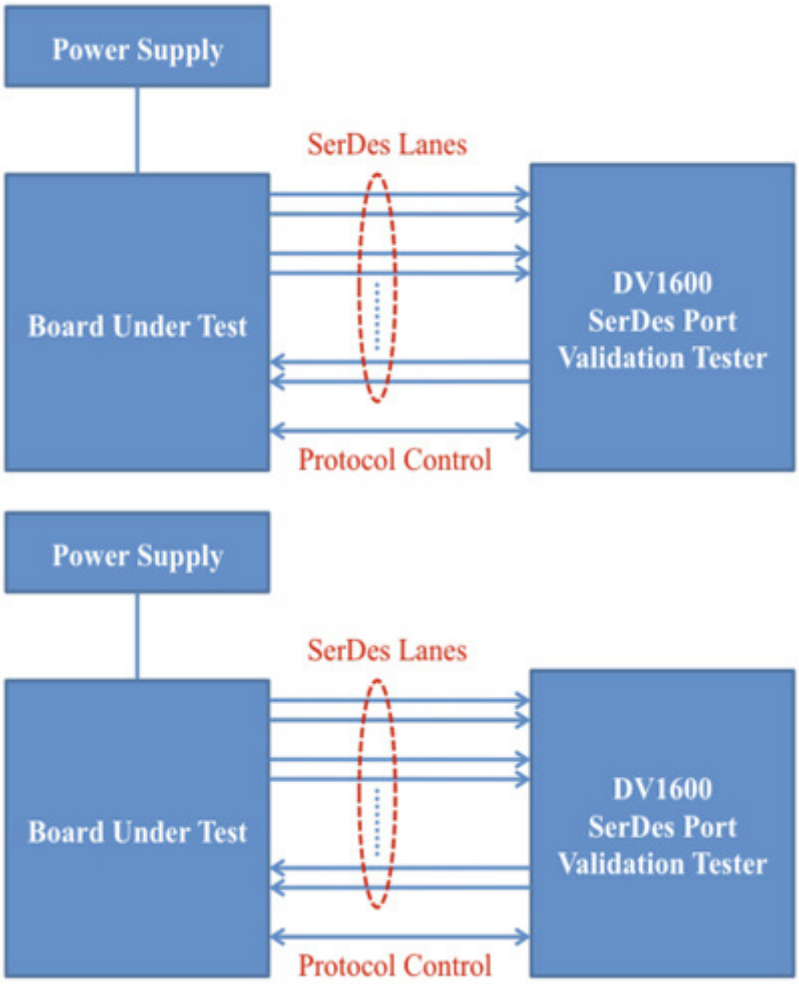
Figure 1 shows three snapshots in the lifetime of a SerDes IP macro from a test and measurement perspective. At the far left, the macro is debugged in dedicated test chips. It is at this phase that most design issues are determined and corrected. Every analog and digital aspect of the macro is analyzed using equipment such as high-bandwidth oscilloscopes, and pattern generators. To the joy of the instrument salesperson, mixed-signal equipment such as noise generators, spectrum analyzers, network analyzers, and phase-noise signal analyzers may be brought to bear as well. At this stage in development, the need for deep understanding of design parameters supersedes any other test feature such as automation or data logging.

In the component bring-up section of figure 1, the SerDes I/O is embedded into a more complete product chip. At this stage, there is some expectation that the I/O will function correctly although signal integrity issues often arise that require detailed debug. For example, a typical problem that occurs is clock feed-through: noise from the digital core clock of the IC makes its way to the output of the SerDes and manifests itself as periodic jitter or data-dependent jitter. Debugging and correcting this issue requires sophisticated equipment just like the rack setup that was described for the first phase.

By now, we have a known good chip with known good SerDes interfaces. The chip is mounted on a system board, and further tests need to be performed (right hand side of figure 1). Just like at the component level, integrating the chip on the PCB with various other components may introduce issues that affect system functionality.

For example, stubs on the PCB traces connected to the SerDes may cause the entire system to fail the target bit-error-rate (BER) specification. Since getting system functionality is the ultimate goal, does this mean that we have to repeat the entire characterization of the SerDes? The answer is no. The SerDes needs to be tested only in the context of making the whole system work. As such, we are much more interested in holistic parameters such as BER than in individual parameters such as random jitter (RJ). We are interested in the interaction of the analog SerDes parameters with overall system performance, not their native values which were already characterized in the first two phases.

In summary, testing at this stage is geared towards proper system operation and achieving maximum link throughput. As far as equipment is concerned, important features are cost, ease-of-use, protocol flexibility, and data logging.
In most of the remaining material, we use the simple, integrated setup shown in Figure 2. In the figure, the board under test is connected directly to a DFT Microsystems DV1600. The DV1600 is a flexible, low cost SerDes validation tester, and we have used it in a variety of system bring-up situations.



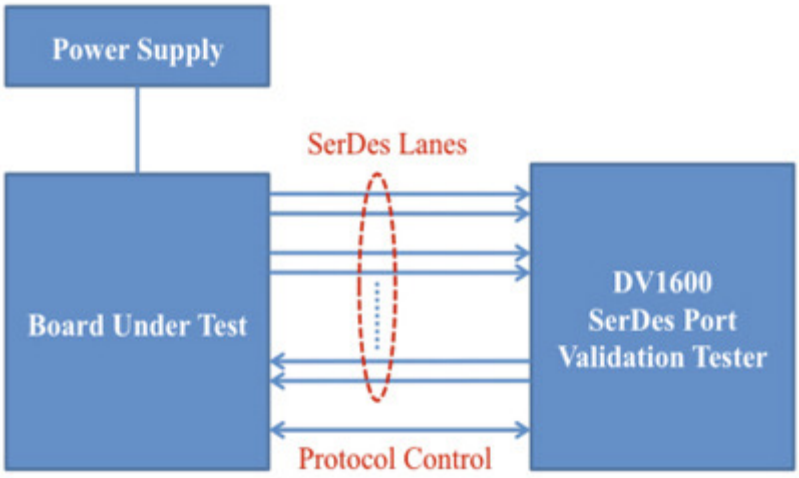


Fig 2: Typical system-level SerDes validation test setup.

Do not forget the bit-error rate

The ultimate goal of any SerDes link is near error-free communication; and the main test for this is a BER test. We often get so involved in analyzing individual parameters such as jitter to the point of paying little attention to the end goal. In reality, at the component level (the first two phases of figure 1), we measure jitter, rise-time, receiver sensitivity, and so on for the purpose of estimating/ensuring BER performance over a wide range of operating conditions. Once the component is on the board, the operating conditions are somewhat set, and it only makes sense to test BER directly. In most board design starts we have seen, doing a direct BER test was one of the fastest ways of identifying and solving SerDes issues.

To measure BER, use a setup like the one in figure 2 and connect the SerDes ports to the instrument ports. For transmitter BER, most instruments automatically align to the device under test (DUT) signal and track its timing, so the setup is simple. For receivers, you have to program the instrument to transmit compliant signals/patterns (see “Do you Speak Protocol?” later in this article).

When measuring BER, do spend the time to accumulate the necessary statistics. Go for a coffee while executing a fifteen minute test. Even better, set up a weekend test with continuous BER accumulation and collect the results the following week. For a 3.125 Gbps SerDes rate, 48 hours amount to 5.4e14 bits. Getting zero errors over this duration implies compliance to the 10-12 error rate requirement with >99.9999% confidence (this calculation is based on confidence intervals, as described in reference 1).

Measuring jitter: do we really need to do it?

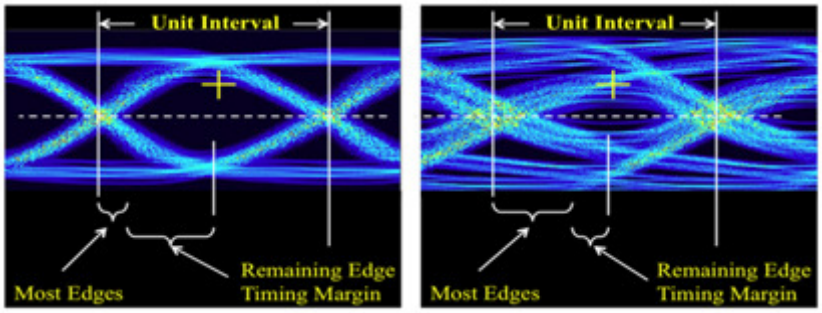
If you asked me 5 years ago, I would have said that you do not need to measure jitter if you pass the BER test and confirm that the link is fine. Back then, PCB technology had plenty of margin relative to overall system specifications. Today, the situation is different for two primary reasons:

- We are running closer to the hairy edge than ever before – less margin
- We often want to tune the link for maximum performance, and the best way to do it is to tune based on jitter (reference 2).

Figure 3 shows two typical eye diagrams to help explain these concepts. If your system is similar to part (a), you have sufficient margin and a BER test is probably a sufficient system-level design screen. In the figure, we classify the regions in the eye as one that has “most edges” and one that has the remaining timing margin. Using jitter terminology, the first region is dominated by deterministic jitter (data-dependent jitter, periodic jitter, etc), and the second region is dominated by RJ due to physical effects such as thermal agitation.

In the system of figure 3(a), there is really no need to characterize RJ down to the single picosecond. RJ is a device phenomenon and is likely to be consistent with previously characterized data from the component

vendor, so let us not worry about re-characterizing it here. On the other hand, the system in figure 3(b) has very little margin: deterministic edge locations occupy most of the UI. So, it is probably desirable to understand how much more timing uncertainty is introduced by RJ. In other words, we are squeezing every bit of performance for this design, so we may need to ensure that system RJ at least matches the component characterization data.



a *b*

Fig 3: Illustration of two typical system-level eye diagrams. In (a), there exists a lot of margin. In (b), there is little margin. Re-characterizing RJ at the system level is more important in (b) than it is in (a).

The example in figure 3(b) leads us to the second reason for measuring jitter (for transmitters and receivers). It is often necessary to tune – by measuring jitter – the link equalization parameters for meeting BER specifications (reference 2).

Specifically, you can use the setup in figure 2 to automatically cycle through all possible pre-emphasis levels in a transmitter and measuring output eye diagrams. The best pre-emphasis setting is selected based on the best measured eye. Similarly, you can loop through all possible receiver equalization settings. Of course, on the receiver side, you generate jitter to optimize the receiver, so it is an indirect measurement.

Instrument clock recovery: when not to use it!

One of the greatest instrument innovations in recent years has been the introduction of clock recovery (CR). It is a terrific tool for de-embedding SSC jitter or for filtering jitter in the frequency domain. So, for the most part, you want to use clock recovery. But, do not be fooled: instrument CR is sometimes too "perfect," and it might not reveal true system issues.

Simply stated, instrument CR usually contains only the CR function whereas a system receiver includes a CR that is coupled to other synchronization/alignment circuitry such as FIFOs. System throughput is determined by the interaction of both components even if the CR was working fine. To illustrate this, refer to figure 4(a) where a real system is measured with CR (left) and without (right). The instrument used was the DV1600, and the board under test had a defective/intermittent fan. Thermal transients in this case caused the SerDes timing to shift wildly, and this is apparent in the odd-looking bathtub on the right hand side of the figure.

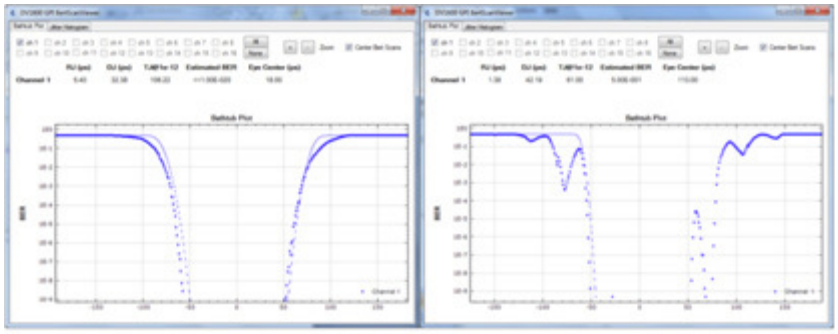
The thermal drifts are tracked well with CR enabled as evidenced by the normal-looking bathtub on the left. Clearly, the intermittent/defective fan was only caught when CR was turned off.

It is important to note that the system in Figure 4(a) might actually boot up properly. But, will it achieve maximum communication throughput? No. The reason it boots up fine is that SerDes protocols embed error recovery mechanisms, so wide timing drifts are intrinsically "concealed" using FIFO circuits and the reliance on synchronization mechanisms (e.g. SKP characters).

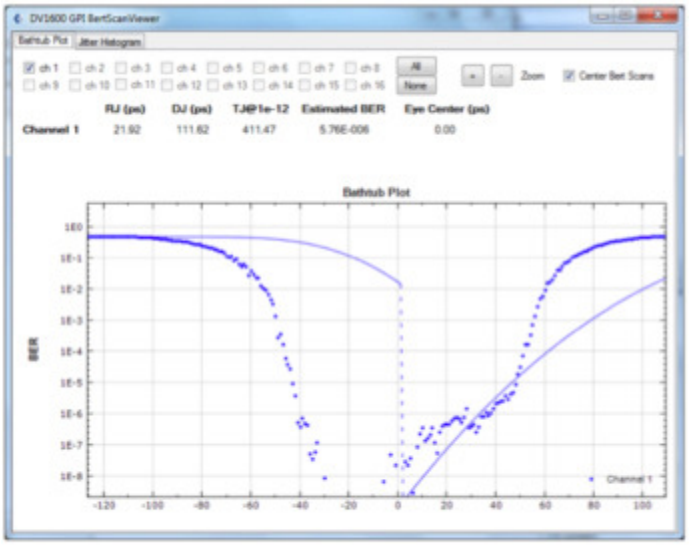
Ironically, it is the same error recovery mechanisms that cause this particular system to be sub-optimal: a large part of the time is spent re-synchronizing (concealing failure) instead of sending useful data. In general, our recommendation is to test with and without CR whenever possible.

Figure 4 shows two more bathtubs that you might encounter when bringing up system boards, both of which are measured without CR. In figure 4(b), a microprocessor periodically varies its workload thus disturbing the power supply slightly, which in turn perturbs the SerDes timing. To the SerDes, these are rare events (10-6 BER in the figure), and the bathtub is the best way to catch them. You can decide whether to do anything about this phenomenon or not based on considerations such as cost, but it is important to know that they exist.

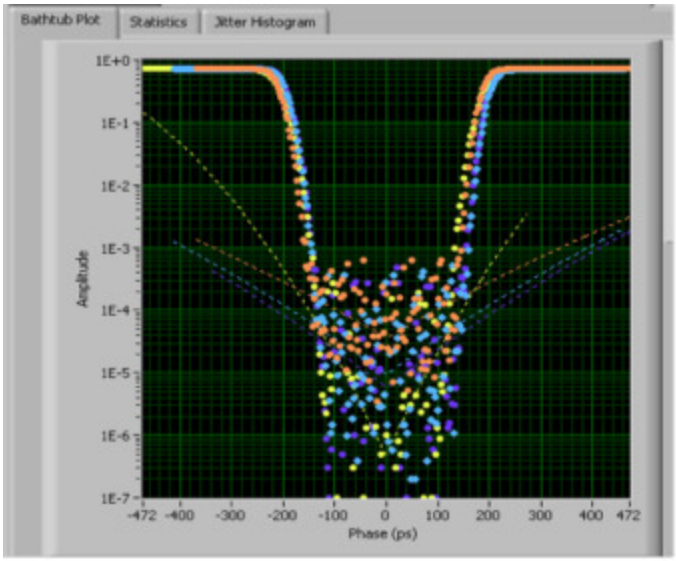
Finally, figure 4(c) shows a typical bathtub that you might encounter if you have an improper crystal oscillator selection. Again, very low frequency errors are encountered in the center of the eye with CR turned off, and again, correcting this is dependent on designer judgment.



(a)



(b)



(c)

Fig 4: (a) Bathtub of a 6.375 Gbps system with a defective fan measured using clock recovery (left) and using constant frequency (right); (b) bathtub of a 6.375 Gbps system that suffers from rare power supply transients. Note that the perturbation lasts from -10ps to +50ps which is a very small timing error. The trouble is that this is a significant portion of the unit interval at this data rate; (c) bathtub of a 2.5 Gbps system that suffers from improper choice of crystal.

Do you speak 'Protocol?'

Last but not least, when the SerDes is instantiated on a system board, you are likely required to talk to it using its native language. The native language depends on the protocol (e.g. PCIe, DisplayPort, etc), and – ideally – you need a tester that can speak the language of the protocol. We have created a protocol-based architecture for the DV1600 for this reason, after having dealt with numerous situations in which a hand-shake was the only means of communicating with the DUT.

Future articles will be dedicated entirely to this topic; here, we briefly describe two scenarios. One is testing Mobile Industry Processor Interface (MIPI) devices. A rather complex hand-shake is required for MIPI before testing can proceed as illustrated in Figure 5. Naturally, we want to get straight to the SerDes tests instead of worrying about device configuration, say, and this is what protocol-based testing is about.

Another example is DisplayPort receiver testing. DisplayPort is unidirectional, which means that BER testing requires a handshake between instrument and DUT: the instrument sends the test pattern (with jitter), the DUT counts errors, and then the instrument interrogates the DUT for the total error count. Without access to a protocol-based instrument, this kind of testing is rather difficult as it might require interaction with hardware and software design teams or tools.

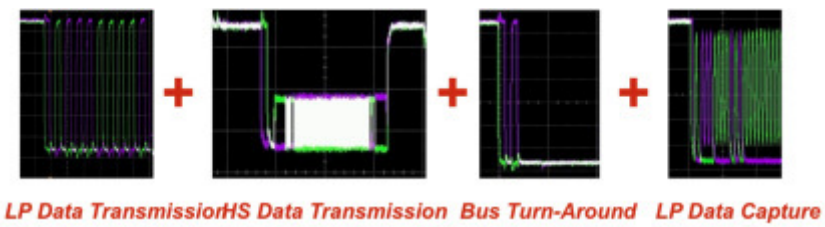


Fig 5: Graphical representation of the sequence of handshakes that need to happen in order to test a MIPI bus. This is a typical receiver test sequence. The challenge is that both single-ended and differential signalling is required, and the instrument is expected to autonomously switch between both modes.

Conclusion

We have described the main objectives of SerDes testing at the system level. We then presented major test and measurement considerations, starting with BER and proceeding to jitter, clock recovery, and protocol awareness.

References

1. Marcus Muller, Ransom Stephens, and Russ McHugh, “Total Jitter Measurement at Low Probability Levels, Using Optimized BERT Scan Method,” DesignCon, 2005.
2. Mohamed Hafed, Justin Moll, “Fabric Test for Mil/Aero Systems: The Cost-Effective Way,” COTS Journal, 2008.

About the author

Dr. Mohamed Hafed is a cofounder of DFT Microsystems http://www.dftmicrosystems.com/ and serves as its Chief Technology Officer. During the company’s early commercialization phases, Dr. Hafed was instrumental to the creation, design, and marketing of the company’s technology and products. In his current role, Dr. Hafed is responsible for product definition and market alignment, and he maintains close relationships with key customers and partners. Dr. Hafed has served on several IEEE committees. He has received several national and international awards for his contributions to mixed-signal test technology, including "Best Paper" awards from both the IEEE International Test Conference and Micronet. He received his B.Eng., M.Eng., and Ph.D. degrees in electrical engineering from McGill University.