By Henrik Liebau, Agilent Technologies Deutschland GmbH

I/O System and Chip Verification in PCI and PCI-X Systems

The complexity of high-performance systems leads to difficult verification.

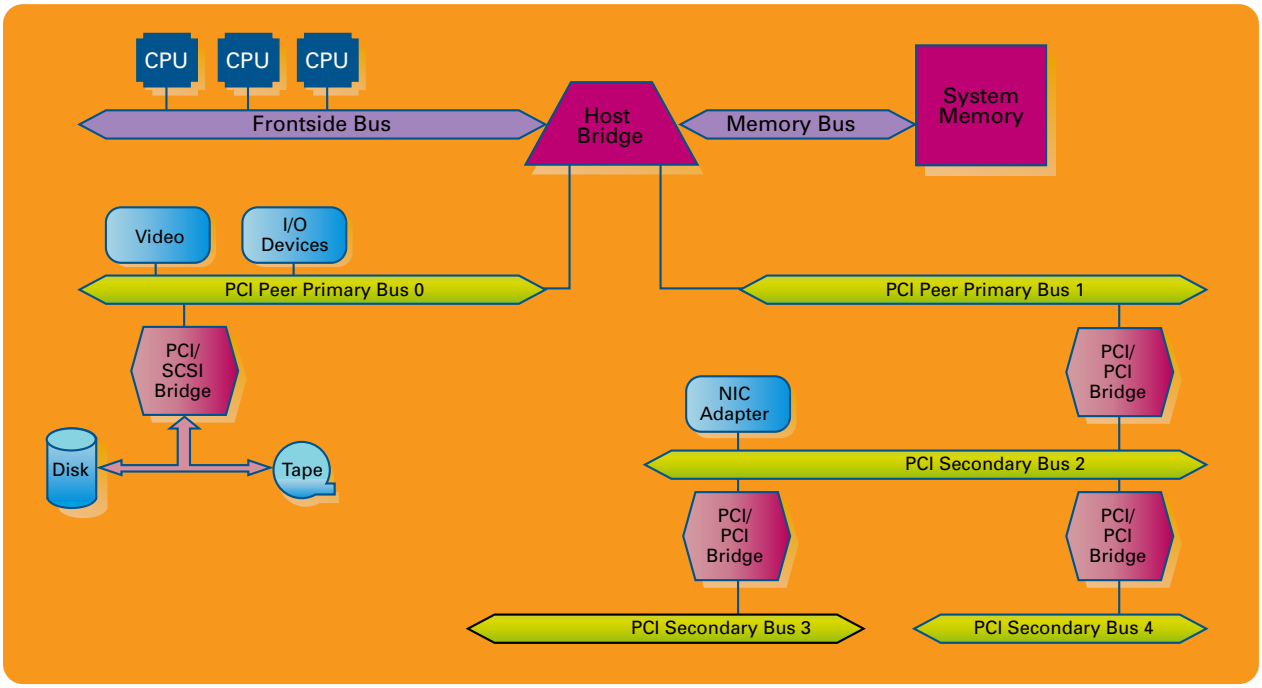

Figure 1. The complexity of modern system structure highlights the problems involved in validation of these systems.

Validation of computer systems and subsystems is becoming increasingly complex as I/O systems and peripherals become more intelligent. The role of data-transfer initiator is being delegated to the I/O systems instead of the CPU, causing data traffic to move in several directions simultaneously and freeing the CPU for data-processing tasks. Increasing bandwidth needs require performance optimization in the subsystems, further increasing the complexity of the whole system.

Validation of these systems is a critical stage in the product development cycle, and is becoming a very complex task. First, you need to verify that all the different subsystems work individually. Then you must verify that the subsystems cooperate when integrated into the system. What is required in this demanding environment are tools that can generate peak-load conditions, show system behavior, stress the system, and validate that the system meets all standards.

A look at modern system structure highlights the problems involved in validation of these systems. Modern servers and workstations comprise various I/O systems, which connect the different peripherals (storage, network, graphics) and low-speed devices (keyboard, serial communications) to the system **(Figure 1)**.

Among these I/O systems, the PCI bus and its designated successor, the PCI-X bus, have established a central role. Their performance, configurability, and scalability make them the choice for all types of peripherals. The ability for bus agents to actively initiate transactions allows intelligent processing to move to the subsystems and relieves the CPUs in the system. For both slot-based and on-board applications, PCI/PCI-X buses play the role of a backbone in the system. Connected to these "backbones" are several other I/O systems, such as SCSI buses. These are connected to the PCI/PCI-X system through bridge devices. PCI/PCI-X systems are emphasized here, but the results are also valid for other systems, including future technologies such as InfiniBand.

Going Parallel

Increasing bandwidth needs call for parallel-processing approaches, so servers and workstations contain several CPUs, numerous storage devices, and several network interface cards. To accommodate these components, multiple I/O buses are built into the system; they are organized hierarchically and/or using peer architecture. The sample system shown in **Figure 1** contains two PCI-X peer buses (0 and 1) and one subsequent PCI bus (2). The combined architecture using both PCI and PCI-X buses is typical, accommodating both state-of-the-art and legacy devices.

With initiators residing anywhere in the system, data traffic patterns become highly unpredictable. This leads to several problems. Traffic in the system becomes non-deterministic, depending on many external factors such as network activity, disk-drive accesses, and CPU utilization. Traffic moves in several directions simultaneously, and with the new split-transaction scheme in PCI-X systems, the traditional initiator/target roles get reversed dynamically. Cache controllers face coherency problems when cached memory is accessed from more than one direction. Bridge devices, which connect similar or different subsystems (such as PCI-X/PCI-X or PCI/SCSI bridges), are confronted with simultaneous requests from all sides.

The insertion of new peripherals into a system brings new problems as new event patterns are introduced and system behavior is changed accordingly. Further adding to this complexity is the possibility that new devices may be introduced in the future, uncovering yet untested scenarios by behaving differently than previously tested devices. Problems that arise from the addition of new peripherals may never occur in day-to-day operation, but could cause a system to fail in extraordinary situations, such as peak-load conditions or rare combinations of events.

Validation Challenge

With all these complexities to cope with, validating systems becomes truly challenging. It is necessary to test systems and subsystems under real-life conditions, while ensuring that corner-cases are covered. It is also necessary to confront the system under test (SUT) with all possible combinations of traffic and to generate peak-load conditions so that the system is stressed to the maximum.

Theoretically, this can be achieved by testing all possible system configurations with every type of hardware that can conceivably be inserted. For a server system this basically means to load it up with all off-the-shelf products available and connect it to a huge, busy network. To guarantee full coverage, this would have to be repeated over and over, with varying configurations. In practical terms, this test approach is not possible due to prohibitive time constraints, not to mention the cost involved. The only current alternative is to decrease test coverage.

Another shortcoming of the brute-force approach is lack of reproducibility: When a problem is found in the validation lab and the SUT needs to go back to the R&D lab, repeatability is an important issue. The cause of a bug can only be found if it can be conjured up consistently with repeatable testing methods.

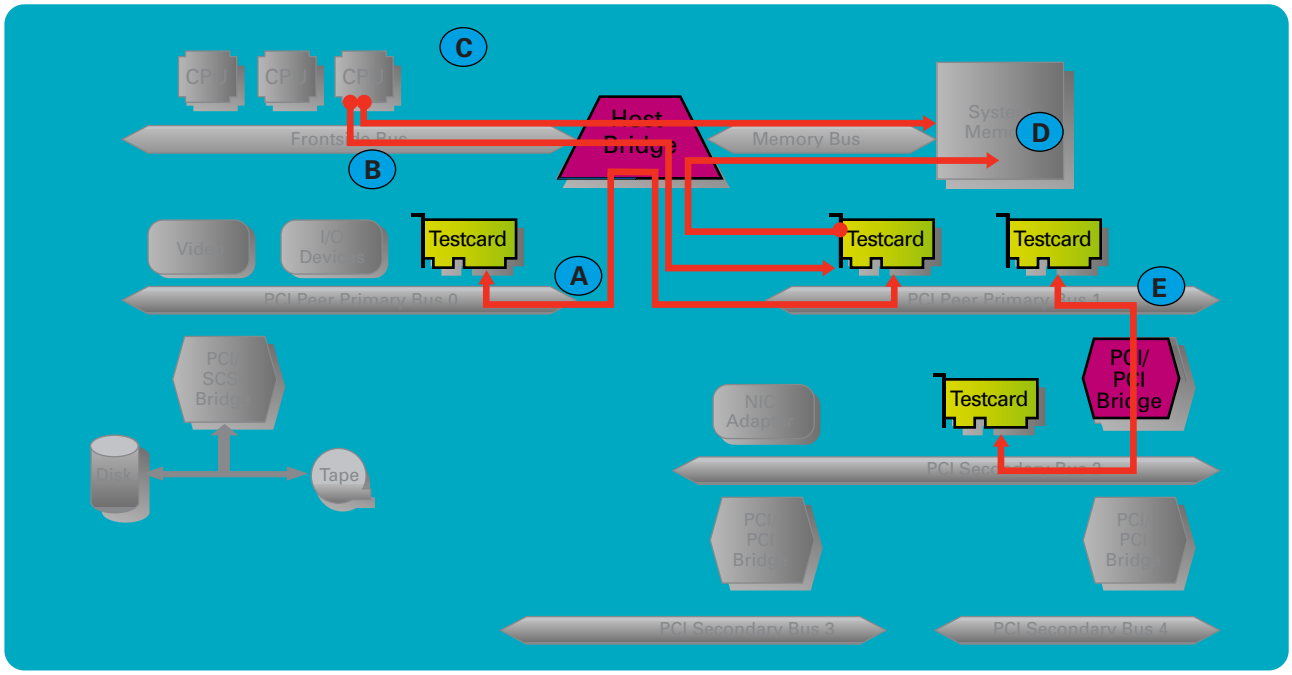

Figure 2. Using test cards, you can generate data traffic along any data path that is accessible by a connector. A bridge can be tested and stressed by placing two cards on either side of the bridge.

Meeting the Challenge

How can a validation engineer ensure that his products are well tested, corner-cases are covered, and the time spent is within reasonable limits? The solution is a combination of test cards and software running on the SUT. To explain this solution, a closer look at the testing cycle for a system is in order. During system integration three basic stages of testing can be distinguished: functional (chip) level, system level, and cluster integration testing.

Functional-level testing, which ensures correct operation of single subsystems, requires short, focused tests. They are handcrafted and designed to test special functional aspects of the device under test (DUT). Functional-level tests need to be adapted for every new device, sometimes even for different versions of a device. Testing may be done in a complete functional system, but the focus lies in testing individual devices.

For system-level testing, data integrity and stability must be ensured along all data paths that are available in the system. The testing must guarantee that, even under peak-load conditions, no single bit gets lost along the way. In contrast to functional-level tests, system-level tests concentrate on load and protocol variation, and can be more standardized. They can be reused in any new system to make sure known problems found in older systems don't occur in the new system. The focus for system-level testing lies more on the data paths rather than on the individual devices.

One step further is the clustering of several systems. Tests on this level are as extensive as the ones at the system level, with the added complexity of intersystem communication. The test-card approach enables testing at all three levels using the same tools.

Test Cards

A test card is a device that is used specifically for testing. It is designed for a specific slot- or cable-based I/O system—PCI, PCI-X, or others—and operates like any other device designed for that system. It can allocate system resources (such as memory space or I/O space), and can act as initiating agent if the I/O system allows it to do so. The test card can generate any type of traffic that is allowed and also can react to any transaction initiated somewhere else.

The test card has an external interface, so it can be controlled with an external controlling host, making it independent of the type of system and the operating system used. When used with an "internal" connection (directly from the SUT), it is possible to coordinate tests using the test card with other test tools or programs.

Using these test-card features, you can generate data traffic along any data path that is accessible by a connector. A bridge can be tested and stressed by placing two cards on either side of the bridge **(Figure 2)**. Programming the test card to communicate directly with a device is a way to test that device. Because the traffic generated by the test card can be pro-

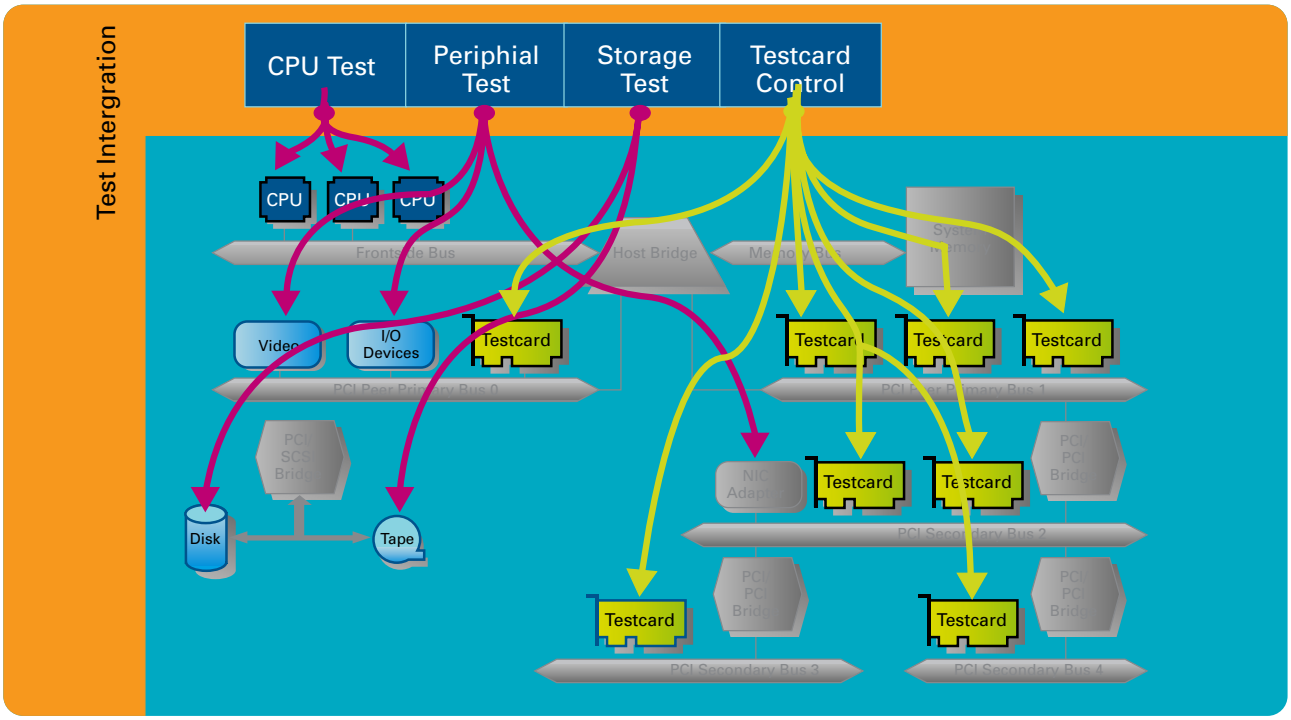

Figure 3. By using appropriate test cards, a single piece of software allows you to easily test very complex systems.

grammed directly, it is also completely repeatable, so generating reproducible results is much easier.

In addition to this active or exercising part of the test card, it is extremely useful if the test card also has analyzing capabilities. These include the ability to monitor the transaction protocol, gather performance metrics, and provide traces of the ongoing transactions should something fail. Together with the exerciser, the analyzer can detect data-integrity problems and protocol violations. It also can generate trigger events for other analyzing devices in the system to provide a "snapshot" of the whole system when a problem occurs in only a part of the system.

The time-consuming swapping of devices can be avoided by characterizing existing devices and incorporating their behavior into a test suite, and then simulating the DUT with a test card.

Window into the System

A test card placed at a central position within the system also can serve as a window into the system. It can read or write registers in system memory or the system's I/O space. It can read or write the configuration data of other devices including bridge devices, and can dump the contents of memory areas. This is possible even when the system itself is no longer operating because the CPU subsystem has crashed or because parts of the system are caught in a live-lock or deadlock situation.

With its exercising and analyzing capabilities, the test card is an ideal tool for linking the test level to the debug level. While trying to track down a bug to its root cause, the engineer can use the same tool for reproducing test results and analyzing them. The R&D and Validation labs can use the same tools. This dual functionality of test cards is also an asset when slots or cables are limited and there is no room to connect two different tools. Before the test card approach, the engineer often had to choose between a test tool and an analyzing tool because of system limitations.

Validation Framework

Some system validation procedures cannot be accomplished by using test cards alone. For example, the processor bus is normally not accessible in the validation stage of the product design cycle. Although it might be feasible to use processor probes or emulators at earlier stages in the design process, using these tools when testing a complete system is uncommon. Conducting tests on the driver or software level becomes impractical when no real devices are used. These problems lead to the combined approach, or "validation framework".

The validation framework handles all aspects of test-card testing—setup, running, and analysis—along with tests that can be written specifically for devices that require CPU interaction. All testing is handled by a single piece of software, which has an application programming interface (API) that allows you to add new tests and to configure existing ones for different needs **(Figure 3)**. User-defined

tests for functional testing on specific devices and more general tests for system integration can be set up and even mixed to “heckle” parts of the system from several sides.

In earlier stages of validation, or in systems with uncommon operating systems, testing can be controlled from outside the SUT using wire connections to the test cards, with only a small stub program running on the SUT. Later, when the environment becomes more stable and mainstream operating systems are used, the same tests can be run and controlled directly within the SUT, along with other tests.

When the system crashes or hangs, the analyzing capabilities of the test cards provide a quick way to check the origin of the problem simply by connecting the test cards from outside and reading the contents of the trace memory. Using the proposed validation-framework approach, multiple errors in several systems in different stages of the validation process—from pre-beta to pre-release—can be found in a matter of minutes. The problems found range from less-important protocol violations on the system buses up to complete system crashes. With the analyzing capability of the test cards, the cause of the problem and the faulty devices can be isolated easily.

Validating complex server and workstation systems has become as complex as the systems themselves. New technologies such as PCI-X add to the complexity. Engineers need tools that help them characterize and validate their system designs quickly. Using a combination of test cards and specific tests combined within a validation framework has proven to be an approach that can significantly reduce testing time and improve the overall reproducibility of the test methods, allowing designers to meet their time-to-market windows.

Verify PCI-X Systems with the Proper Tools

PCI and the higher-performance PCI-X technology provide a challenge when developing high-performance systems. These technologies are complex and difficult to verify and validate. The Agilent Technologies E2929A and E2922A allow you to easily control PCI-X-based chipsets, servers, server clusters, or other high-performance systems. You can then create repeatable test scenarios that will properly evaluate your implementation.

The E2929A PCI-X exerciser/analyzer is fully compliant with the 133-MHz PCI-X specification. It is a single-slot card that features complete PCI-X state analysis, real-time protocol checking, and real-time performance measurement. The card offers a fully programmable PCI-X master/target with completer and requester capabilities. It also provides data memory, data generation, real-time data comparison, and programmable configuration space.

The E2922A PCI-X Master/Target Test Card is a dedicated PCI-X exerciser that provides a fast and predictable way to set up the PCI-X traffic, verify PCI-X protocol compliance, and verify the target chipsets where multiple test cards per test setup are needed.

Agilent Technologies also provides a variety of measurement and analysis solutions for PCI and PCI-X, with the 16700A/B series of logic analyzers providing multiple-bus analysis, cross-domain analysis, and timing analysis.

For more information, check 4 on the reply card.