HEWLETT PACKARD

# Meeting the Challenge of Computer System Validation

White Paper

HP Böblingen Verification Solutions

**Included in this paper:**

# Taking the Random Out of Random

Having established itself as the standard for PC bus systems, PCI (peripheral component interconnect) provides an open platform allowing for the growth of very complex computer systems.

**Plug and Play—Are Open Architectures Too Open?**

The open environments where end users can plug in any combination of adapter cards and run any type of software provide the basis for very complex systems. To prevent built-in obsolescence, board manufacturers have to engineer into their systems a future they actually don't even know. They need to allow for an ever growing multiplicity of hardware, software, and firmware components.

**Growing Diversity ...**

... *of Components* Typical servers or high-end PCs, for example, are often built on a multi-bus and/or multi-processor architecture. System manufacturers have to ensure that their configuration of motherboard, processor type, PCI chipset, memory system, and bus topology runs with any combination of NICs (Network Interface Cards), network software, SCSI adapters, hard disks, etc. the customer might choose to plug in.

... *of Operating Systems* Operating systems also are no longer Rocks of Gibraltar in the stormy seas of innovation. They themselves are subject to development and change. In order to remain compatible for the future, the multiplicity of components must be matched with any variety of different operating systems and applications.

... *of Applications* Also, at the level of mission critical applications (high-end servers, mainframe, industrial control, and communication hubs) fact needs to be separated from fiction: The proverbial flight simulator is no longer sufficient for system validation. Other applications such as standard office systems or test programs verifying file transfer via LAN or SCSI adapters have reached just as high a level of complexity and need to be considered in testing.

There is only one reliable prediction for the future: Complexity will keep growing. Indeed, after many years of validity, Moore's law—predicting that the complexity on a single chip would double every 18 months—may well be overtaken by the arrival of PCI 64 bit, 66 MHz.

Consequently, reality might far exceed the boundaries of what the most learned predictions calculate. Manufacturers of system boards and adapter cards alike have no freedom to cut corners on the validation of their products. And validation methods as well, need to keep track with this rapid development.

## What Makes Systems Crash Makes Systems Stable

From their day-to-day experience any computer user is witness to the fact that the more a system has added to it in terms of hardware and software, the higher the probability that it crashes—often more than once a day. A system that doesn't is what one might call stable. And that's what system validation is all about: If a system is stressed under real life conditions and remains stable, then it is likely to be stable.

## Goals and Methods of System Validation

Validation teams have several important goals ...

... *ideally* They need to make sure that the system works correctly with all supported operating systems, in all configurations that a user can put together, and under worst-case conditions. They also need to convincingly communicate test results.

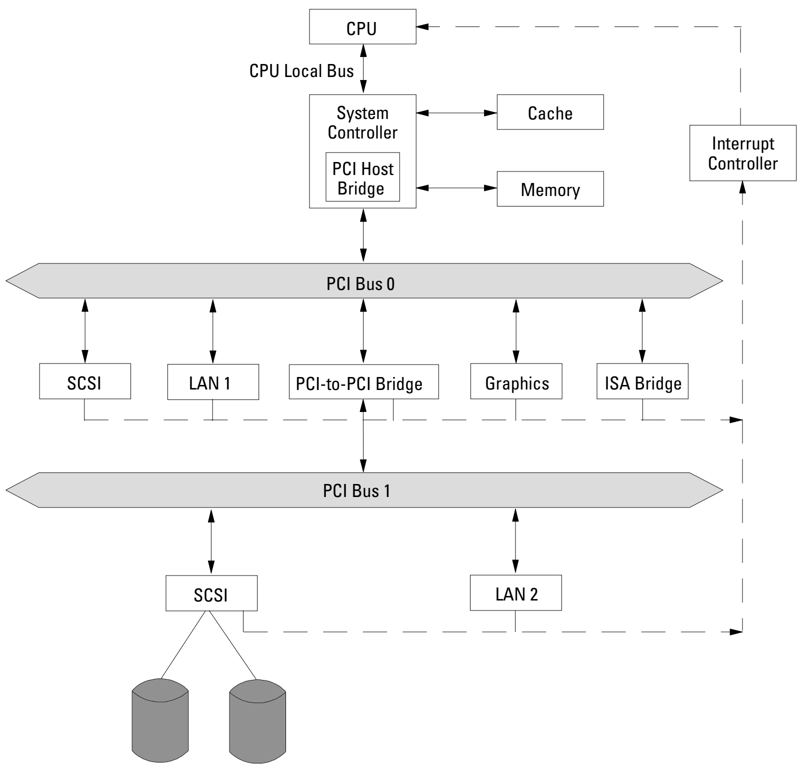
... *realistically* They have to achieve the optimum test coverage within a short period of time. Test design and test results should be measurable. This means, that engineers search for the right level of coverage in order to gain maximum confidence.

One method of increasing test coverage is stressing the system. Increasing stress conditions significantly intensifies system validation tests, increases test coverage, and forces the early occurrence of critical system problems.

... *forcing compromises at the point of diminishing return* Increasing test coverage results in increased confidence, but raises costs in time and money. This will inevitably lead to a cost versus reliability trade-off: **The point of diminishing return**.

## System Validation and its Present Limits

The figure below shows an example of the complex architecture of high-end PCI computer systems. The system is built of components like the memory system (memory, cache, and memory controller), the system controller (PCI Host bridge connecting processor, memory system, and PCI bus), PCI buses and bridges between them, the interrupt system as well as master and target devices. It is important to emphasize the mutual nature of the relationship between all components.



System validation needs to ensure that all components and the system as a whole behave as planned—even under stress conditions.

## Definitions of Stress

What is generally understood as stress for a computer system? If an office application automated by macros is doing its job while the CD player is running, and data is transferred via the network interface card in parallel—that means stress for the average computer.

### Application-Level Stress

The system bus is still where the bottleneck occurs: A PCI-based computer system is stressed by heavy traffic on the PCI bus. Not much change from the olden days. This means that loading the bus with lots of traffic coming from different sources should produce good testing conditions. To generate this type of stress validation teams use an application oriented approach by running so-called “Hot Mock-Up tests”.

### Hot Mock-Up tests—Too Hot to quantify?

Lots of cards are plugged into the system in order to check whether the system or card under evaluation is capable of dealing with these components.

The cards are usually taken from a stock consisting of the usual off-the-shelf products, especially of the notorious “bad guys” among PCI compliant products. These cards are known to often cause problems within a system despite being PCI compatible. Using different cards allows creation of lots of traffic—lots of different traffic.

Each of these cards are designed to be PCI compatible. However, the PCI specification only defines the boundaries telling designers what they can and cannot do, but still leaving them a great latitude of design possibilities. Each designer can choose a different way of implementing compliance in his PCI device. Running only “Hot Mock-Up tests” will never lead to optimum test coverage and confidence as there are too many possibilities.

### Running the Paces

The system is booted over and over again to check its behavior during this critical phase. While a system is booting there is a lot of stress: Disk access, memory access, LAN traffic, etc.

Applications like flight simulators, other games, user applications, and benchmark programs are run to generate heavy traffic. Simultaneously, other test programs are run to check for errors (read-write-compare tests, for example).

These methods of generating application-level stress have one benefit, they are very realistic. They allow for identification of components showing poor performance and they highlight other problems, for example crashes caused by data integrity problems.

### Hard to repeat and hard to report

As shown above, most of the stress factors can be generated by means of “Hot Mock-Up tests”. This may be a very realistic way to go but on the other hand it is rather time-consuming, hard to repeat, and hard to report.

Obviously, in order to overcome the limitations of “real life” in system validation, another method must be found.

Even though “real life” in system validation cannot be substituted it can be supplemented. The solution is to use test cards incorporating the full scope of possible stress factors and the means to control them.

## Optimizing Test Coverage—An Impossible Equation?

Under these circumstances, optimizing test coverage seems to lead to an impossible equation. True, the goal of optimizing test coverage is not to reach a fictitious maximum coverage but to approach the right level of coverage for a maximum confidence. Nevertheless, the following questions must be asked and they must be answered:

- How to control test coverage?
- Will a selection of sample test devices do?
- How to find suitable test devices?
- What test coverage has been reached?
- Where to set the limit?

Using the conventional application-level approach does not answer all these questions.

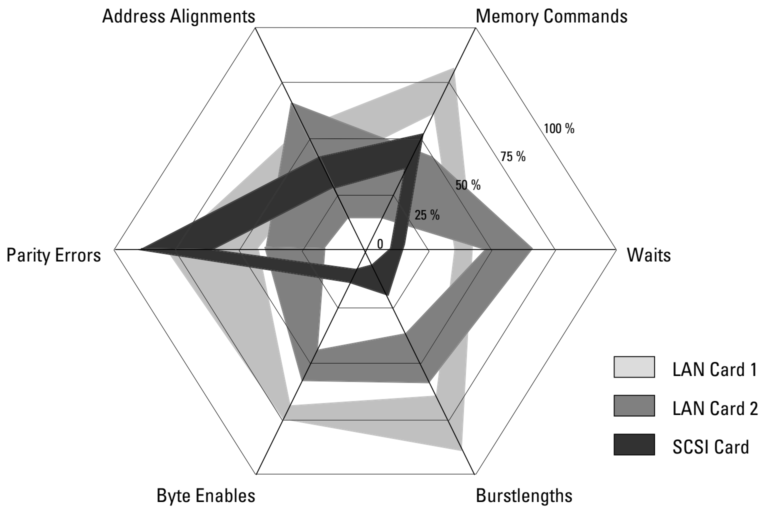
## Missing Transparency

In many cases such a lack of answers grows out of the simple fact that there is no certain way of finding them. In fact, the list of questions is much longer:

- How large are the gaps not covered by the used cards?
- Has the entire scope of the cards' behavior been covered?

Different NICs plugged into the system, for example, might use a wide variety of different burstlengths for transferring their data. But the likelihood of ever testing enough NICs to expose a system to all burstlengths likely to occur, must be set extremely low.

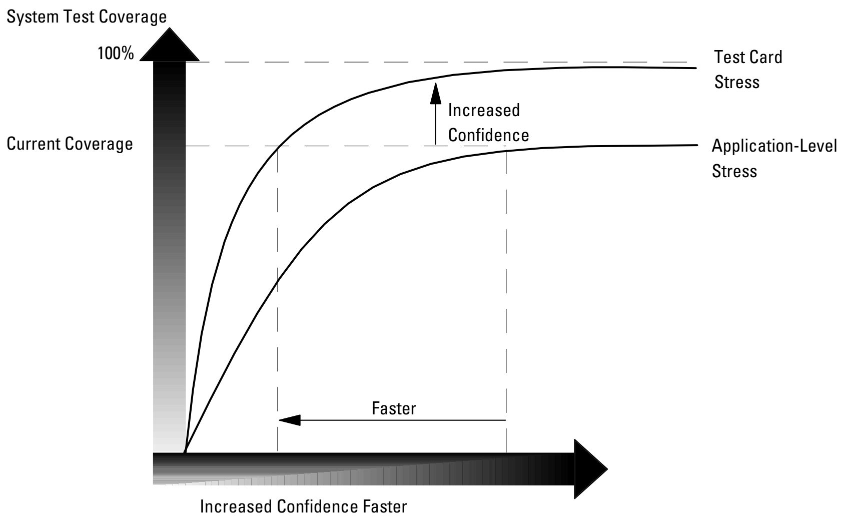
The following figure shows the coverage—and the gaps—that might be reached using three sample test devices.



Often, a test runs for hours and days before it fails. And when a problem occurs—eureka!—it is hard to tell which component caused it under what circumstances. Thus the list continues:

- What happened before the failure?
- How can this be repeated? Will it *ever* happen again?
- Which of the apparent problems is the most important one?

There are no easy answers, and often there are none at all. Instead, the questions seem to multiply exponentially with each step in computing development.



## When I'm Good—I'm Good?

When a system has reached a certain stability, system validation engineers face the necessity to place a quantification label onto that stability. But how can they turn up with any figures on something still plagued by all these open questions.

### Measurability—Prerequisite for Communication

## Coverage—only half the Story

One of the key factors in generating confidence is communication. One may have the best product and will never sell a single item of it, if no one knows just how good it is. Without the conviction of the quality, however, not even the best salesman can really convey quality. System validation engineers, therefore, need to be convinced themselves of the quality of their product. And they need a measure as a means of communicating such a quality.

## Increasing Demands on all Engineering and Performance Issues

A clear consequence arising from all these requirements and limitations is the need for a new and/or supplemental approach to the business of system validation. This must clearly be an approach leading beyond the point of diminishing return encountered by conventional methods. The demands on quality continue to increase. So will the pace of development and the need to get products to market very fast. The pressure on profitability will rise as well. Machines and people will need to perform more efficiently as the margins for error keep shrinking.

## PCI Bus Window Into the System

Fortunately, the PCI bus not only is the platform for evolving variability and performance but also supports the growing needs of system validation. As a window into the system it can serve as an excellent pathway for testing and validation purposes, providing all the insight required.

In the following, the new test card approach to system validation is introduced as a stand-alone solution and as an addition to be integrated within the conventional method of testing a system by means of user applications, test programs, and off-the-shelf devices.

## The New Approach

**Increased Coverage with Test Cards**

The test card approach intensifies validation tests by generating variable levels of background traffic, by emulating typical peripheral traffic patterns to stress arbitration, memory controller, bridges and system interrupts, and thus creating controlled stress conditions.

**Programmable: Controlling Stress in a Deterministic Way**

The programmable hardware and software of such cards will allow for the generation of controlled stress. And only controlled stress can really achieve best test coverage. After the test is completed, there is a verifiable report showing that the system was exposed to all programmed stress situations.

**Able to Simulate Real-Time Traffic**

Test cards have another advantage: they can be used to simulate the behavior of any other card and to generate any kind of traffic. Considering the example discussed above concerning the need to cover all burstlengths it is a logical step to use the programmability of the test card to systematically check all burstlengths.

## Another Definition of Stress

By using a test card plugged onto the PCI bus, it is possible to access each system component directly via the bus, and fully control the stress factors. When stressing a system this way, hidden design faults will show up earlier than under normal testing conditions.

**Memory System Stress**

The memory system may be stressed by concurring memory accesses using different parameters. Examples for these parameters are memory commands (PCI provides different commands for memory access), burstlength and bandwidth, memory and cacheline alignment. This is what these “bad” cards mostly do to your system. They use unusual (but valid) combinations of these parameters, so-called corner cases.

**Traffic Load and Protocol Stress**

This is what happens when you have lots of cards in your system and run those test programs. There is a lot of load on the bus, with different master devices requiring bus access.

This causes waits, retries, release request etc., thus stressing the bus arbitration and the bus masters. Again “bad” cards show unusual behavior, thus adding corner cases.

**Interrupt Stress**

Adding interrupts to a loaded system provides even more stress, and shows how the system reacts when device interrupt requests cannot be serviced in time.

Exposing a system to these stress factors, also shows whether the system’s error handling mechanisms operate correctly, detecting and reporting parity errors, system errors, etc.

## Leaving Nothing To Chance

As it turns out, the equation calculating the effort for development and the result in quality using test cards produces additional good news for manufacturers.

As validation can become a concurrent process in engineering rather than a static point at the end of each phase of development, it will continuously provide insight into design behavior. It will therefore reduce the system validation time while significantly increasing test coverage.

### On Schedule With Confidence

Using the test card approach, the knowledge and experience of development engineers will come to fruition on a much larger scale. By programming the wealth of their know-how into a validation setup they will get a more substantial feedback on their personal experience. They will feel much more in control of the process because they are in control of the process.

In the race for ever faster—everything, actually—the demands on quality are gaining importance as well: *Does the performance hold up?*

In such a situation it is important to have the chance for validation even before all parts of the system are available. Using test cards, PCI-based designs can be tested early on in the development process leading to the earliest possible detection and pin-pointing of system-critical design problems.

**Adding stability to performance**

The logical consequence of these capabilities is the improvement of performance and stability while still in the design and engineering stages.

### Controlling the Stress Factors

Controlling stress in a deterministic way will help to quickly identify bottlenecks like slow components causing slow data transfers, and components causing data integrity problems and system hang ups. As a result, these components may be exchanged early on.

To ease this task random permutations of PCI protocol parameters can be performed within customer-defined ranges to add a specific protocol stress. Such randomly structured tests beat any linearly structured tests.

Finally, to avoid start-up delays, test cards provide ready-to-use—but customizable—test programs to stress and monitor all system components. These programs include, for example, protocol checker, memory stress tests, bandwidth consumer tests.

## No Cutting Corners but Cutting Costs

**Future Built in**

Since the longevity and the life expectancy of a product are often dependent on the so-called future compatibility, test coverage has to be brought beyond the present point of just enough—for now.

The test card approach to validation uses the PCI bus to force an application oriented traffic to stress the system. Based on the knowledge how such traffic affects a system, this approach brings in a process-oriented view on testing, allowing to manage the increasing complexity. It has been shown invariably that such kind of stress can bring up hidden design problems more quickly than any user application with randomly-generated stress.

## Deterministic Testing

**Knowing what will be tested**

Using test cards and their accompanying software as the vehicle for validating systems and components will yield what has been missing up to now. Not only the facts are revealed (and provoked) but also the age old question: why? will be answered

- *Observing* the progress or deviation under programmable and controlled circumstances leads to a comprehensive understanding of the reasons behind any failure.
- *Logging* the transactions of all system processes delivers the data necessary to efficiently go about root cause analysis.
- *Analyzing* the data collected will narrow down or pinpoint the source of any problem or deficiency.
- *Repeatable Validation* supports communication between teams—and with customers

The reports delivered by test cards are required to pass valuable information on to the R & D teams responsible. These may in turn analyze the logs, repeat and verify the tests, and come up with the necessary corrective action.

# Stressing Systems Rather Than Stressing People

**You Get What You Planned**

Programmable hardware and software make it possible to generate controlled stress in order to achieve an optimum of test coverage. This allows to custom tailor any test exactly to the specific requirements. Once the test is run, there is no uncertainty left if the system was really exposed to all stress situations programmed. All relevant gaps have been closed.

### Repeatable Faults

Programmable test cards are able to generate real-world traffic conditions and to apply stress beyond what is possible with current off-the-shelf devices. This allows for validation tests to be entirely deterministic so that when any problem occurs it can be repeated.

This deterministic testing, however, shouldn't be carried to extremes, because no time should be spent on problems that will never occur in the user's environment. Only realistic stress conditions are worth being tested.

### Flexibility

Reaching the right level of test coverage requires the tools used to be very flexible. By directly programming a test card by means of an application programming interface (C-API) the engineer has full control over the traffic generated by the card and over the protocol parameters used for that traffic.

### Standard Setting Usually Sufficient

**If it Works—Why Fix it?**

In the system validation environment, where the system-under-test is typically stable enough to run an operating system, the test card can simply be plugged in and do its job. No standard validation setting must be eliminated, no fundamental test re-design is necessary.

In a typical first step, therefore, the card might be used only in parallel to the existing test programs to run a protocol check. This requires no extra effort, but increases test coverage.

### Programmable Additions

**Integrates Within Existing Test Environments**

Next, the card may be programmed to add background load to the bus. This significantly increases stress conditions. For this task, test cards provide ready-to-use functions—again, no extra effort.

Finally the test card's full flexibility can be exploited using the C-API. Test programs may be written checking special aspects of the system or device being tested. Multiple test cards can be used to thoroughly put individual subsystem components through their paces, like the memory system, a PCI-to-PCI bridge, etc.

For testing the systems boot behavior the cards configuration space may be controlled up to the point of being completely invisible to the system under test by requiring no configuration space at all.

### Easy Integration with Existing Test Programs

Using the C-API, the tests running on the test card can easily be integrated into existing test programs. User-written benchmarks may be run in conjunction with system stress test functions provided by the cards software tools. You can guarantee coordinated action from CPU and PCI, for example, to stress memory controller and cache.

## Protects Investment

The bottom line on these integrative characteristics and features of test cards is the bottom line. Existing investment not only is protected, it gets a new lease on productive life. The accompanying modular software and hardware ensures the flexibility required for specific solutions not rigid ones.

Therefore, starting with a test card approach doesn't mean replacing existing test environments, it means enhancing them.

## Summary

**No Magic Wand**

Clearly, there can be no magic wand to resolve all quality issues in today's rapidly paced development. But the test card approach coupled with extensive programming libraries represents a milestone enhancement to existing test environments

Considering the process-oriented view required to manage the increasing complexity of today's computer systems and the growing demands on system validation the test card approach can not really obviate the need for the good ol' "Hot Mock-Up" run today. But they can make a significant contribution beyond what "Hot Mock-Up tests" can do for system validation. The turning point quality they bring into the business of validating computer systems arises from their predictability.

**Predictability**

Their major benefit is that they create a predictable validation environment with controlled test conditions, allowing to exactly protocol and reproduce the testing conditions that may have caused any fault or—conversely—represented a mark of quality and performance.

**Verifiability**

With its reporting capabilities, a test card delivers valuable information that can easily be fed back into the debugging process within R & D. This creates a new test culture, as it re-defines the role of the validation teams within the product development cycle. They can now provide detailed information, not just annoying bug reports.

**Flexibility**

The flexibility provided by test cards and its associated software tools and libraries allows to enhance test coverage step by step, without sacrificing existing test environments.

**Profitability**

Finally, each problem detected during system validation in the course of development—rather than at the customer's site—avoids image loss and reduces warranty costs.

Engineers working with a predictable validation tool are more confident of their product and, thus, deliver *better products faster*. And nothing will rub off on customers' expectations and satisfaction as the company's own confidence—throughout the ranks.

## References

This White Paper is based on

- information from the following lectures:
  - "Efficient Use of PCI" held by Frank Hady (Platform Architecture Labs, Intel Corporation) at the PCI+, 1997
  - "How to Find Boundary Condition Problems in Your PCI Bus Interface" held by Jeffrey K Witt (Symbios Inc.) at the DesignCon98
  - "PCI System Validation" held by Thomas Dippon (Hewlett Packard GmbH) at the PCI+, 1997
- working experience with the HP E2920 Computer Verification Tools, PCI Series
- information from the HP E2920 Computer Verification Tools, PCI Series brochure, p/n 5965-4723E
- information from the HP E2920 PCI Performance Multimedia CD, p/n 5965-5882E
- information from the HP Digital Verification Tools Home Page, http://www.hp.com/go/dvt