

Surviving State Disruptions Caused by Test: A Case Study

Kenneth P. Parker, Agilent Technologies
Shuichi Kameyama¹, Fujitsu Limited
David Dubberke, Intel Corporation

Abstract

The practice of initializing a board or system for testing purposes is not an exact science, but rather, pursued empirically and with an increasing risk of undesired side effects. It has been suspected that Boundary-Scan testing can cause such side effects. This paper provides a case study of such a board where a detailed root-cause analysis was performed. Some issues are identified that justify adding features to IEEE 1149.1 that will facilitate safe, fast and effective initialization of a board or system, to get it ready for testing and to leave it in a safe state upon completion of testing.

1 Disclaimer

There is a Working Group for IEEE 1149.1 that is revising the standard and could implement some of the ideas presented in this paper (see [IEEEWG]). The content of this paper may not reflect the group's final thinking and results. Opinions stated throughout this paper are those of the authors.

1. Also with Ehime University, Matsuyama, Ehime, Japan.

Copyright © [2011] IEEE. Reprinted from IEEE

Paper 5.2

First presented at ITC Week, International Test Conference, September 18-23, 2011

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Agilent's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to: pubs-permissions at ieee dot org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.



Agilent Technologies

2 Introduction

A paper [Park10], presented at the 2010 IEEE International Test Conference described the effects of the “Lobotomy” problem [Park03] which is a side-effect of using the IEEE 1149.1 test standard [IEEE01], [IEEE03] for testing a complex printed circuit board (PCB). This (perhaps overly colorful) term refers to the fact that Boundary-Scan testing starts with silicon devices in “normal” mode of operation. Then, all the Boundary-Scan devices are switched to “test” mode where the I/O pins are suddenly under the control of internal test register(s), with no synchronization with the activities that were in operation on the PCB. This can be disruptive of the board’s normal function, and the function of system logic inside each IC that is participating in the test. The term “lobotomy” refers to the radical medical practice of surgically disconnecting the frontal lobe of a person’s brain, which produces stark, irreversible cognitive and personality changes in its recipients. On a PCB, Boundary-Scan tests disconnect device I/O pins from their internal intelligence suddenly and without any warning. A problem could exist when the Boundary-Scan test completes and the device TAP controllers proceed to the Test-Logic-Reset state where I/O pins are reconnected to the internal device logic. It can be difficult to predict the outcome of the devices switching from “normal” mode to “test” mode and back to “normal” mode. The purpose of the [Park10] paper was to analyze what features could be added to IEEE Std 1149.1 that could be used to control unpredictable side-effects of passing to and from the test mode.

This paper describes an actual case of a board that was reported by Fujitsu to have some unexpected side-effects that occurred during, and after, Boundary-Scan testing. Engineers at Fujitsu did detailed research of what was actually happening, which is described here. Then, the current thinking of the IEEE 1149.1 Working Group is examined to see how it would help solve these problems.

3 A Case Study

A board designed by Fujitsu, containing several devices with Boundary-Scan, was giving trouble to test engineers. The board is a personal computer mother Board and uses a device referred to as the "PCH" (Platform Control Hub) [Intel10]. See Figure 1 and its block diagram in Figure 2.

The PCH collects a number of functional responsibilities that are needed for implementing major systems into a single device. The PCH contains numerous I/O interfaces (LAN, PCIe, SATA, USB etc), and supports the "Advanced Configuration and Power Interface" (ACPI) [Intel] which is an open standard specification for device configuration and power management. The ACPI is a combination of hardware and firmware that manages "Power States", "Device States", "Processor States" and "Performance States". The PCH controls some of the onboard voltage regulators by turning them on or off, to support the different power states defined within the ACPI specification. Terms such as "Sleep", "Suspend" and "Hibernation" are associated with the system feature set.



Figure 1. A Fujitsu board that was analyzed.

3 A Case Study (continued)

The PCH and other intelligent devices are actively involved in real-time power management. Fail-safe checks and watchdog functions are active to keep voltages within intended limits during normal operation.

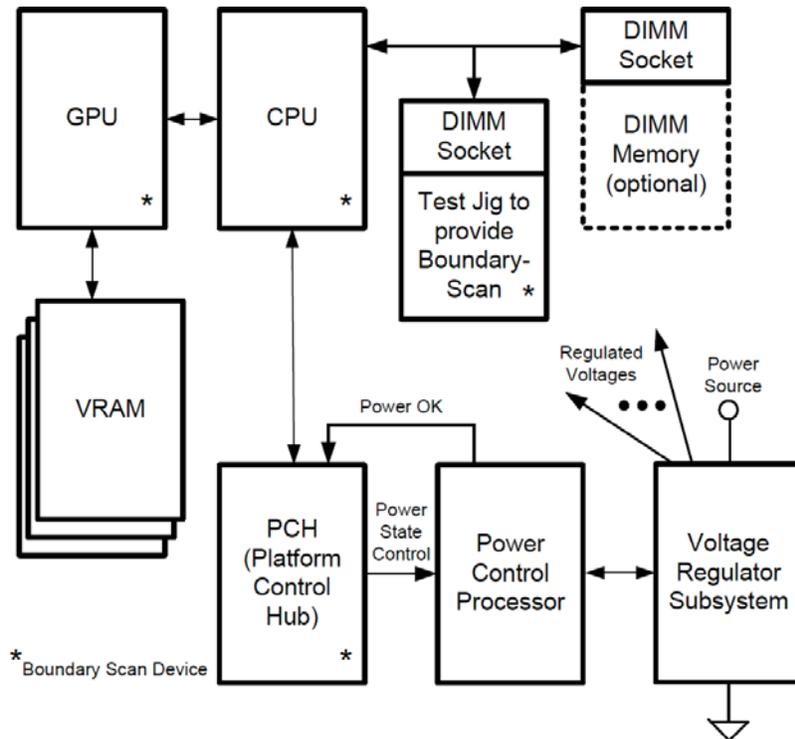


Figure 2. Block diagram of the board.

As seen in Figure 2, the PCH and the power control processor work together to support the different powered states by controlling which voltage regulators are on or off that feed other board logic. During test, the PCH outputs need to be properly conditioned so the voltage regulation remains stable. But, what happens when a test is completed and the PCH is no longer in test mode?

Today's devices are busily managing their own internal power consumption. They can turn major sections of circuitry on and off as needed. This can translate into higher and lower power current requirements over time.

Test engineers may not be aware of all the complex power functions, and, the board may not be a complete system during testing. For example, DRAM modules may not be present during testing, which means the PCH and/or other major devices may not be able to execute certain code functions. Add to this the "lobotimization" of the system circuitry inside these devices during Boundary-Scan test, and there are reasons to suspect side effects could occur.

3 A Case Study (continued)

The board in question for this case study had the situation where active power management features can be interfered with by Boundary-Scan testing. The Fujitsu board test engineers had a test strategy based on powering up the board without all of the DIMM modules in place. In place of the memory modules, they inserted “Test Jig” devices, which would complete the Boundary-Scan circuit between the jig and the board, to gain coverage of the DIMM sockets. Then a Fujitsu-designed Boundary-Scan tester would be used to test the interconnect circuitry for manufacturing defects. The test was segmented into two phases (which are greatly simplified in this discussion); each would start and end in the Test-Logic-Reset TAP state, and each would use PRELOAD to set up pin data in boundary registers before switching to EXTEST. The first test checked interconnections between the ICs with Boundary Scan. The second test exercised the GPU/VRAM interface with EXTEST, while the other ICs were “normal”. But in some cases, the second test failed to execute properly. It was determined that some of the on-board power voltage regulators would turn off between the executions of EXTEST in the two Boundary-Scan tests. It was not easy to understand why this was happening, so laboratory analysis was needed.

Fujitsu engineers, with guidance from Intel, worked to discover what was happening. As long as the PCH had EXTEST control of the I/O through the boundary register, the power rails would remain on. When the PCH boundary register was no longer in control of the I/O by transitioning to the Test-Logic-Reset state or executing the PRELOAD instruction, the power rails could turn off. During the Boundary-Scan test, board and signal conditions had been met to trigger a watchdog timer running in the PCH to perform a warm reset that would power-cycle the voltage regulators under PCH control. This happened in parallel with the interval between the completion of the first Boundary-Scan test, and the start of the second Boundary-Scan test devoted to testing the GPU and VRAM. This second test was thus interrupted by a power cycle which would confuse it. The Boundary-Scan test controller, running from its own power supply, would not be aware of this event and would not be able to make sense of the erroneous results that were received.

Two solutions were developed to resolve this behavior. First, they could load one of the memory DIMMs with real memory, so that the PCH would behave differently and not decide to cycle the power. In this case, the watchdog condition was not triggered. A second method was to reverse the order of execution of the two tests. This tested the GPU/VRAM portion first without disturbing the PCH. Then the main interconnect test (including the PCH) was executed. After completion of the tests in this order, a power cycle was tolerable.

Needless to say, understanding this problem required skilled investigation, and this subverts a value proposition of Boundary-Scan, that boards should be testable without detailed knowledge of their system function.

So, what could be done to solve this problem? This type of question is currently under review of the IEEE 1149.1 Working Group since the 1149.1 standard has not been updated since 2001. The next sections introduce some concepts that provide a “toolset” that test engineers can use to engineer a dependable, safe state for board testing to use both between and after multiple tests are run.

4 Initializing ICs for Testing

There have been some major changes in ICs since 2001. Many of these changes have been driven by device complexity. There are many devices available with “programmable” features, which include programmable I/O behavior. For example, an I/O pin can be given a “family” personality so that it appears to be a 3.0 volt pin, or, maybe a 1.8 volt pin, or maybe one of several others. High-speed I/O pins may be given protocols to follow, such as PCI Express, or SATA. Their differential voltage swings may be tuned (selected) to full, or lesser swings such as 75% or 52.7%. Their operating frequency may also be selectable, say between 100 and 125 MHz. These selections may apply to disjoint sets of pins, which implies configuration memory behind these groups of pins.

Boundary-Scan testing does not require frequencies or I/O protocols to be selected. The I/O should be divorced from these properties so that I/O pins will behave in the very simple ways that the 1149.x family of standards requires. Device I/O electrical characteristics (for logic levels and AC/DC coupled signals) must match connected devices, so that the devices can transmit/receive data between each other successfully. The electrical characteristic configuration must be accomplished before Boundary-Scan testing switches pins from normal mode to test mode, or, tests may fail due to electrical incompatibility rather than because of manufacturing defects on the board.

Complex devices also manage their internal power consumption. They can literally turn on/off various internal subsystems. This helps with systematic goals such as enhancing battery life or reducing heat buildup. The second goal may be important for testing, since many boards may not have their thermal solutions in place when they enter the manufacturing test stage, as they can be mechanically bulky and incompatible with test fixtures. But Boundary-Scan test will require that power be applied to the board, so we do not want to have excessive heat generation during this time. Thus, it would be desirable for devices to have accessible shutdown modes where heat-generating subsystems are held in a “safe and cool” state [Park10] for the duration of testing. The feature would effectively alert a device that testing is about to commence and for that time the device should be quiescent.

The 1149.1 Working Group has devised two new instructions to support this, named INIT_SETUP and INIT_RUN, which were described in [Park10]. The INIT_SETUP instruction gives access to an INIT_DATA register, which contains the data needed to select I/O pin family specification data for voltage levels, communication protocols, pin frequencies and so on. One important type of data would be the parameters needed for 1149.6 Test Receivers [IEEE03], so they can be adjusted to properly detect signals from upstream ICs that may themselves have selectable protocols. Note the INIT_SETUP instruction is non-invasive; it does not interfere with normal IC operation.

4 Initializing ICs for Testing *(continued)*

The INIT_RUN instruction is used to drive internal state machines at the TCK frequency, to set up internal requirements for the test, or more generally, to shut off certain functionality that may be generating noise interference or dissipating heat. This can get the IC into a “safe and cool” mode so that testing can proceed without concern for compromised cooling often seen during testing. INIT_RUN is invasive to IC function, and the I/O pins behave as if CLAMP is in effect. This means a suitable I/O state must be in place (using PRELOAD) before this instruction is executed.

The Working Group has encountered examples of ICs where the amount of data needed in the INIT_DATA register would be many hundreds, even thousands of bits long. A given instance of such an IC may need a download of bits very specific to that IC’s location on a board. Other ICs of that same type may need their own, customized downloads as well. This data may consist of numerous, relatively short fields of data that select I/O voltage levels per pin, for example. Thus, new BSDL (see [IEEE01], Annex B) constructs are being developed to describe such fields within a longer register and give them meaningful names. Similarly, data patterns that would be loaded into these fields will also be given mnemonic names. Then, a register field like “PLL-A” might be assigned a bit pattern with the name “PLLOff”, meaning a phase-lock loop A is turned off, as part of the test setup. Similarly, a given I/O interface may be programmed to “SATA” voltage levels since these are compatible with the voltage levels expected on the neighboring IC those pins are connected to.

An implication of this is each instance of an IC that contains an INIT_DATA register may need a unique “side file” of initialization data. The Working Group envisions GUI-based tools that assist the test engineer with setting up each side file. These tools would work using meaningful register names and data mnemonics, and hide the details of which bit locations get loaded with what bit patterns. IC suppliers could also supply default side files used as a starting point, such that the test engineer would just make adjustments to its content for the context that IC was found in on the board.

5 Test-mode Persistence

In [Park10], the concept of a “Ready-for-Test” modal state was introduced. This modal state would be set up by the INITIALIZE process (the use of INIT_SETUP and INIT_RUN instructions) and the resultant set up state would be “persistent” even when the chained TAPs were sent to the Test-Logic-Reset (TLR) state. That meant several Boundary-Scan tests, each devised to be “stand-alone” could be run in any order once the chained devices were initialized by INIT_SETUP and INIT_RUN. The fact that the internal device logic had been decoupled from the I/O pins would no longer be of concern since the I/O pins would not be reconnected to their internal logic at random points of time between tests. The Ready-for-Test modal state was a place to “park” the chained device TAPs between tests. There is also an assumption here that the I/O states used during a test are consistent and safe; that is, they are not setting up problematic conditions such as bus driver fights.

The 1149.1 Working Group has since this time developed a new (and optional) mode of Boundary-Scan operation that adopts some of the Ready-for-Test concept. It is the idea of “Test-mode Persistence”.

Boundary-Scan tests put chained device I/Os into test mode (typically after using PRELOAD to set up test states) where their I/Os are completely under control of the Boundary register content. They do this with instructions like EXTEST or CLAMP. But when the chained TAPs pass through the TLR state, these instructions are replaced with BYPASS (or IDCODE) which are not test mode instructions. The I/O pins then revert to being connected to the internal device logic, which were lobotomized by entering test mode. The results of this reconnection are unpredictable. Note that when several independent Boundary Scan tests are run sequentially, each may use its own PRELOAD sequence to set up for the use of EXTEST, and while this is being done, the I/Os are also reconnected to system logic.

The CLAMP_HOLD instruction is envisioned to eliminate this reconnection of I/O pins at TLR and when non-test instructions are loaded. This is to say, the I/O pins would remain under control of the Boundary register content (test mode) even if the Instruction register was loaded with BYPASS or IDCODE upon entering the TLR state. This was at first a problematical behavior in the Working Group’s thinking since for over 20 years, the non-test instructions were “normal” mode instructions that did not control the I/O pins. The question was how to implement this new behavior. For example, one could imagine the TAP state diagram as having new states. But this was considered too radical a change. So, a different approach was taken, that of an optional parallel state diagram called the Test Persistence Controller (TPC). This retains the standard 16-state TAP diagram, but the TPC governs how particular non-test instructions behave, by giving them a bimodal definition.

The TPC has two states, called Test-mode Persistence On and Test-mode Persistence Off (see Figure 3). Two new instructions² are used to toggle between the On/Off states; the CLAMP_HOLD instruction and the CLAMP_RELEASE instruction.³

2. *These names may change up until the release of a new 1149.1 standard.*

3. *The working group has also considered a third arc from the On to the Off state called the “BYPASS Escape” arc, used as a fail-safe mechanism. This detail is omitted from this discussion for simplicity.*

5 Test-mode Persistence (continued)

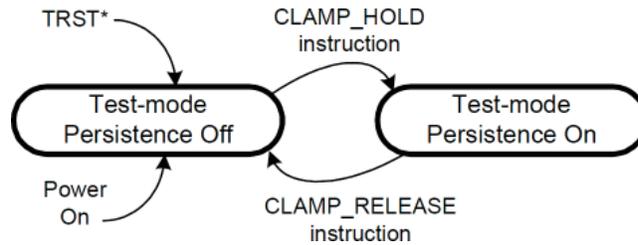


Figure 3. The Test Persistence Controller state diagram.

Note that passing through the TLR TAP state does not cause a change in the TPC state. On assertion of the TRST* pin, or when an On-Chip POR (Power-On Reset, if present) is active, the TPC will move to the Off state.

To achieve Test-mode Persistence at the I/O pins, we need to first look at how a typical output pin is controlled by a Boundary register cell. An example is the classic “BC_1” data cell shown in Figure 4. (Its output is sent to a pad driver.) Common to all such Boundary register cells is a “Mode” signal derived in the TAP from instruction decoding. (The Mode signal value is shown in the table in the figure versus instruction.) When Mode=0, then the output pin is connected to the Boundary register cell input, which would come from the device’s system logic. When Mode=1, then the output pin state is controlled by the current content of the Update flip-flop (R2), and thus the I/O is disconnected from the system logic.

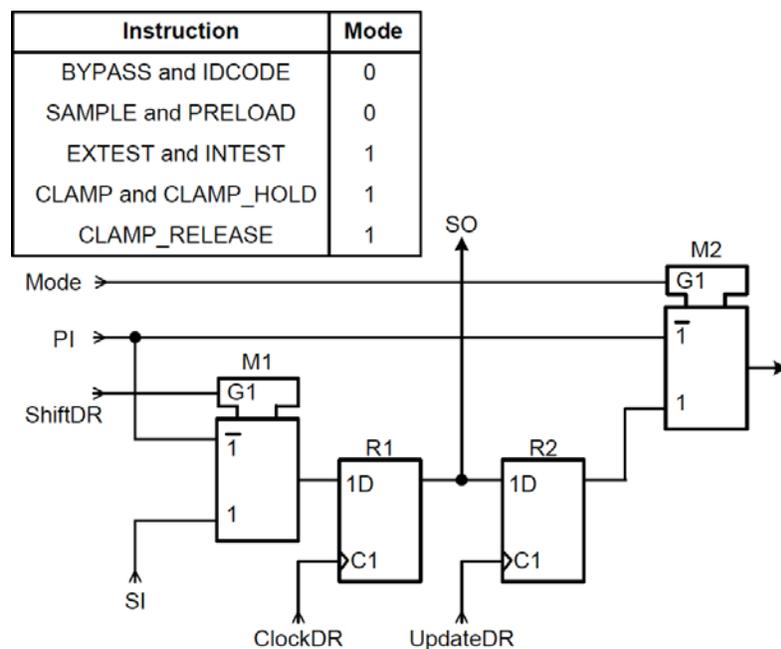


Figure 4. Boundary Register Cell BC_1 from Figure 11-30 of [IEEE01].

5 Test-mode Persistence (continued)

Similar cells of the control-and-observe type may be used for input pins. There, the Mode line determines if the system logic is connected (or not) to device input pins. Note that when an instruction that has Mode=1 is replaced by an instruction with Mode=0, the data to/from the Boundary register will be switched out and the I/O pins are reconnected to the internal device logic. This automatically happens whenever the TAP passes through the TLR state, as this loads the Instruction register with BYPASS (or IDCODE). However, the concept of Testmode Persistence requires that this mode change does not occur, when persistence is "On". The Mode signal must be modified to support "persistence on". (See the CH-Mode signal in Figure 6.)

The 1149.1 standard also defines output pin enable control cells. These can also be classic "BC_1" designs. But, at the option of the device designer, such control cells can be set (or reset) to the state that disables the driver, when the TAP passes the TLR state.

An example of such a design is shown in Figure 5, where Reset* is used to set (or reset) the Update flip-flop (R2). The Reset* signal is generated in the TAP and is asserted at the TLR state. Normally, such an action may be justified if at the end of a test (where the TAP goes to the TLR state) the device designer would like drivers to be automatically disabled. However, the concept of Testmode Persistence requires that this driver disabling does not occur. The Reset* signal must be modified to support "persistence on". (See CH-ControlR Reset* in Figure 6.)

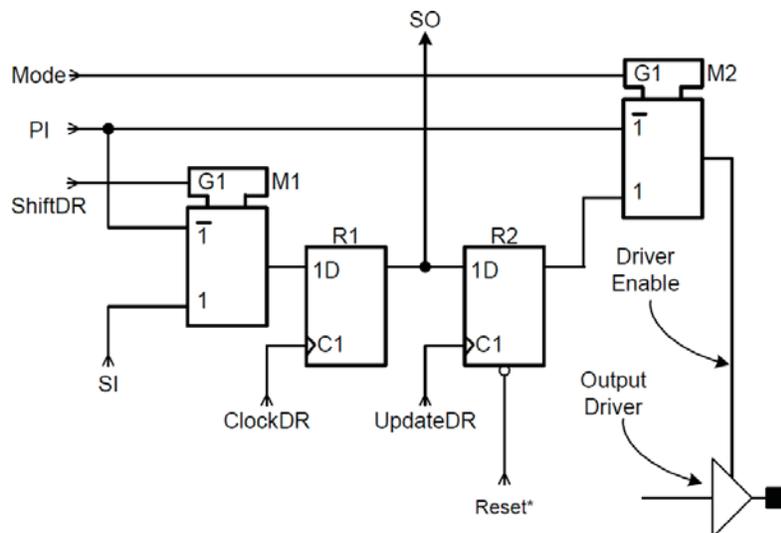


Figure 5. Boundary Register Cell with pre-settable Update Flip-Flop (ControlR).

5 Test-mode Persistence (continued)

The Test-mode Persistence controller has two states, so it can be implemented with a single flip-flop.⁴ A possible implementation is shown in Figure 6.

The TPC can be located in the TAP where its input signals are readily available. It generates the two modified Mode and Reset* signals, for distribution to the Boundary register. Note that with this change, the number of signals distributed across the IC is unchanged.

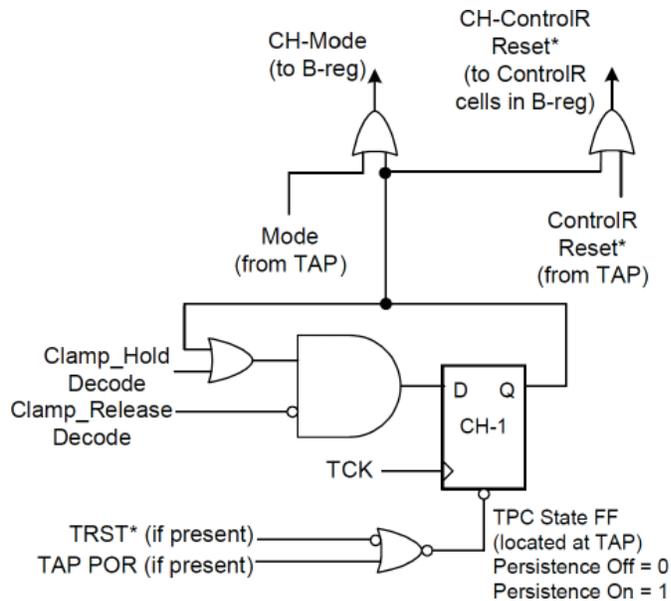


Figure 6. An implementation of the Test-mode Persistence Controller.

4. If a device designer is concerned about "single-event upset" (SEU) errors that could lock a device into test mode, the design can be modified to contain multiple, redundant flip-flops that are "voted" on to determine if persistence is on/off.

5 Test-mode Persistence (continued)

The TPC memory flip-flop CH-1 is cleared (Off) at power up or when TRST* is asserted. It loads a '1' (On) when CLAMP_HOLD is the updated instruction. Once set, only an assertion of TRST* or the loading of CLAMP_RELEASE can clear this bit. Also notice that since the normal Mode line generated by the TAP in response to the currently loaded instruction only changes in the Update-IR TAP state, changes in the TPC state are not observable until either CLAMP_HOLD or CLAMP_RELEASE are displaced by another instruction, since both of these instructions specify Mode to be '1'.

All the standard instructions generate a Mode signal, and effectively, this is overridden to '1' when Test-mode Persistence is implemented and turned on. This generates new I/O pin behavior which is summarized in Table 1 at the end of this paper.

The main goal of Test-mode Persistence is to maintain the last driven states of the I/O when a test completes, so that either the board stays in that quiescent state until power down, or, some new test that is to begin sometime later will start from that state. Today, when multiple tests are run in sequence, there is a gap of time between the end of one (where the TAP went to the TLR state) and the point in the following test where test mode is re-entered, typically after a PRELOAD sequence. During this time the device I/Os are connected to lobotomized device logic, with unknown results that are difficult to predict.

This indeed was the root cause for trouble experienced by the test case at Fujitsu. If (at a minimum) the PCH chip had a Test-mode Persistence Controller, the CLAMP_HOLD instruction could be used to freeze the signal pins that control the power regulation in a stable state. This state would be maintained after the first test completed and the next test was getting ready to run. For the case observed at Fujitsu, maintaining static states on the PCH would prevent the autonomous cycling of the power subsystem.

6 TAP-based IC Reset

The IC_RESET instruction, termed “RESTORE” in [Park10], is a “TAP-based” method for asserting the Master Reset function pin on a device. This pin could be otherwise blocked by an instruction (like EXTEST) or by the Test-mode Persistence mode being set to “On”.

The Working Group has also decided to address the reality that many ICs contain Intellectual Property (IP) that may come from different sources. For example, a piece of IP that performs on-chip memory BIST might be included in several places in an IC’s design. Such an IP block might have its own reset signal. Thus, IC_RESET is envisioned (at this writing) to address a Reset-Select register. This register can be used to enable and toggle any reset function that comes from a device input pin, or those utilized internally by IP blocks.

The goal is to allow individual control of external or internal reset functions, using only the TAP pins. This can be used to ensure that on-chip subsystems are reset during testing activities, or, they can be blocked from being reset by test related signaling. This gives test engineers another tool to suppress unpredictable behaviors that result from testing, or, the completion of testing.

7 Conclusions

The “lobotomy problem” that occurs when Boundary-Scan testing completes can disrupt board behavior. This disruption can lead to events that can disrupt subsequent test activities. This was demonstrated by Fujitsu engineers who did a painstaking root-cause analysis of a complicated board that would sometimes exhibit strange behavior during subsequent Boundary-Scan testing. It was proven that earlier Boundary-Scan testing created conditions in the power conditioning circuitry that would interfere with later testing.

New Boundary-Scan instructions, INIT_SETUP, INIT_RUN, CLAMP_HOLD, CLAMP_RELEASE and IC_RESET that are being studied by the IEEE 1149.1 Working Group [IEEEWG] can provide the tools needed to remove these types of problems. The case given here shows how this can be done.

To summarize, if a suite of Boundary Scan tests is created for a board that contains ICs that have this new Test-mode Persistence capability, then test generation software has several options:

- At the end of the *first* test, load the CLAMP_HOLD instruction. This freezes the last EXTEST pattern in place on the I/Os. But, this test should always be run first when the system is in “normal” mode, such as after power-up. The clamping behavior persists through all the remaining tests, although each may complete with a different clamped pattern on the outputs;
- or, at the end of *each* test, load CLAMP_HOLD after the last EXTEST pattern is completed. This freezes the last EXTEST pattern state in place. Tests in the suite can then be run in any order, or some may be deleted at will.
- The *last* test in the test suite can load the CLAMP_RELEASE instruction. This releases the I/Os of all devices that were in the persistent state. This may be acceptable if the resultant state is safe, or, a re-boot of board function is the predictable result;
- or, a specifically engineered routine using the CLAMP_RELEASE and IC_RESET instructions can be appended to the *end* of the suite of tests. This routine effectively re-boots the board. The test engineer will need to investigate which IC is “the master” with respect to re-booting and target that device with IC_RESET;
- or, cycle the power to cause a general re-boot. Power cycling is guaranteed to remove the persistence of test mode and should invoke the appropriate re-boot behavior designed into the board. Power cycling, however, can be time consuming, which may cause a throughput concern.

8 Acknowledgement

The authors would like to thank James Grealish at Intel for his help in explaining the silicon in question. Fujitsu engineers Masayuki Baba and Manabu Keyaki were instrumental in performing the root-cause analysis of the interactions uncovered in this case study. Finally, the 1149.1 Working Group has worked diligently for over a year on these questions. We believe these new tools will be important contributions to realizing safe and effective Boundary Scan tests, into the future.

9 References

- [IEEE01] "IEEE Standard Test Access Port and Boundary-Scan Architecture", IEEE Std 1149.1-2001
- [IEEE03] "IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks", IEEE Std 1149.6-2003
- [IEEEWG] IEEE 1149.1 Working Group website which contains meeting minutes and a private draft area: <http://grouper.ieee.org/groups/1149/1/>
- [Intel10] "Intel® 5 Series Chipset and Intel® 3400 Series Chipset", Intel Corporation Document Number: 322169-003, June 2010
- [Intel] <http://www.intel.com/technology/iapc/acpi/>
- [Park03] "The Boundary-Scan Handbook", 3rd Edition, Parker, K. P., Kluwer Academic Publishers (now Springer), Boston, MA, 2003
- [Park10] Parker, K. P., "Surviving State Disruptions Caused by Test: The 'Lobotomy Problem'", Proc. IEEE International Test Conference, paper 19.2, Austin Tx, Nov 2010



Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Channel Partners

www.agilent.com/find/channelpartners
Get the best of both worlds: Agilent's measurement expertise and product breadth, combined with channel partner convenience.

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas

Canada	(877) 894 4414
Brazil	(11) 4197 3500
Mexico	01800 5064 800
United States	(800) 829 4444

Asia Pacific

Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Other AP Countries	(65) 375 8100

Europe & Middle East

Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700*
	*0.125 €/minute
Germany	49 (0) 7031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
United Kingdom	44 (0) 131 452 0200

For other unlisted countries:

www.agilent.com/find/contactus

Revised: June 8, 2011

Product specifications and descriptions in this document subject to change without notice.

© Agilent Technologies, Inc. 2011
Published in USA, November 4, 2011
5990-9430EN

