

# Pre-Silicon Validation of Hyper-Threading Technology

David Burns, Desktop Platforms Group, Intel Corp.

Index words: microprocessor, validation, bugs, verification

## ABSTRACT

Hyper-Threading Technology delivers significantly improved architectural performance at a lower-than-traditional power consumption and die size cost. However, increased logic complexity is one of the trade-offs of this technology. Hyper-Threading Technology exponentially increases the micro-architectural state space, decreases validation controllability, and creates a number of new and interesting micro-architectural boundary conditions. On the Intel® Xeon™ processor family, which implements two logical processors per physical processor, there are multiple, independent logical processor selection points that use several algorithms to determine logical processor selection. Four types of resources: Duplicated, Fully Shared, Entry Tagged, and Partitioned, are used to support the technology. This complexity adds to the pre-silicon validation challenge.

Not only is the architectural state space much larger (see “Hyper-Threading Technology Architecture and Microarchitecture” in this issue of the *Intel Technology Journal*), but also a temporal factor is involved. Testing an architectural state may not be effective if one logical processor is halted before the other logical processor is halted. The multiple, independent, logical processor selection points and interference from simultaneously executing instructions reduce controllability. This in turn increases the difficulty of setting up precise boundary conditions to test. Supporting four resource types creates new validation conditions such as cross-logical processor corruption of the architectural state. Moreover, Hyper-Threading Technology provides support for inter- and intra-logical processor store to load forwarding, greatly increasing the challenge of memory ordering and memory coherency validation.

---

® Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

™ Xeon is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

This paper describes how Hyper-Threading Technology impacts pre-silicon validation, the new validation challenges created by this technology, and our strategy for pre-silicon validation. Bug data are then presented and used to demonstrate the effectiveness of our pre-silicon Hyper-Threading Technology validation.

## INTRODUCTION

Intel IA-32 processors that feature the Intel® NetBurst™ microarchitecture can also support Hyper-Threading Technology or simultaneous multi-threading (SMT). Pre-silicon validation of Hyper-Threading Technology was successfully accomplished in parallel with the Pentium® 4 processor pre-silicon validation, and it leveraged the Pentium 4 processor pre-silicon validation techniques of Formal Verification (FV), Cluster Test Environments (CTEs), Architecture Validation (AV), and Coverage-Based Validation.

## THE CHALLENGES OF PRE-SILICON HYPER-THREADING TECHNOLOGY VALIDATION

The main validation challenge presented by Hyper-Threading Technology is an increase in complexity that manifested itself in these major ways:

- Project management issues
- An increase in the number of operating modes: MT-mode, ST0-mode, and ST1-mode, each described in “Hyper-Threading Technology Architecture and Microarchitecture” in this issue of the *Intel Technology Journal*.

---

® Intel and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

™ NetBurst is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

- Hyper-Threading Technology squared the architectural state space.
- A decrease in controllability.
- An increase in the number and complexity of microarchitectural boundary conditions.
- New validation concerns for logical processor starvation and fairness.

Microprocessor validation already was an exercise in the intractable engineering problem of ensuring the correct functionality of an immensely complex design with a limited budget and on a tight schedule. Hyper-Threading Technology made it even more intractable. Hyper-Threading Technology did not demand entirely new validation methods and it did fit within the already planned Pentium® 4 processor validation framework of formal verification, cluster testing, architectural validation, and coverage-based microarchitectural validation. What Hyper-Threading Technology did require, however, was an increase in validation staffing and a significant increase in computing capacity.

### Project Management

The major pre-silicon validation project management decision was where to use the additional staff. Was a single team, which focused exclusively on Hyper-Threading Technology validation, needed? Should all the current functional validation teams focus on Hyper-Threading Technology validation? The answer, driven by the pervasiveness, complexity, and implementation of the technology, was both. All of the existing pre-silicon validation teams assumed responsibility for portions of the validation, and a new small team of experienced engineers was formed to focus exclusively on Hyper-Threading Technology validation. The task was divided as follows:

- *Coverage-based validation* [1] teams employed coverage validation at the microcode, cluster, and full-chip levels. Approximately thirty percent of the coded conditions were related to Hyper-Threading Technology. As discussed later in this paper, the use of cluster test environments was essential for overcoming the controllability issues posed by the technology.
- The *Architecture Validation (AV)* [1] team fully explored the IA-32 Instruction Set Architecture space. The tests were primarily single-threaded tests

(meaning the test has only a single thread of execution and therefore each test runs on one logical processor) and were run on each logical processor to ensure symmetry.

- The *Formal Verification (FV)* team proved high-risk logical processor-related properties. Nearly one-third of the FV proofs were for Hyper-Threading Technology [1].
- The *MT Validation (MTV)* team validated specific issues raised in the Hyper-Threading Technology architecture specification and any related validation area not covered by other teams. Special attention was paid to the cross product of the architectural state space, logical processor data sharing, logical processor forward progress, atomic operations and self-modifying code.

### Operating Modes

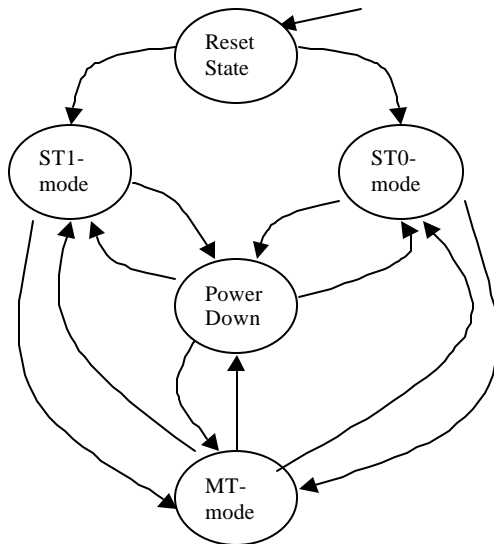
Hyper-Threading Technology led to the creation of the three operating modes, MT, ST0, and ST1, and four general types of resources used to implement Hyper-Threading Technology. These resources can be categorized as follows:

- **Duplicated.** This is where the resources required to maintain the unique architectural state of each logical processor are replicated.
- **Partitioned.** This is where a structure is divided in half between the logical processors in MT-mode and fully utilized by the active logical processor in ST0- or ST1-mode.
- **Entry Tagged.** This is where the overall structure is competitively shared, but the individual entries are owned by a logical processor and identified with a logical processor ID.
- **Fully Shared.** This is where logical processors compete on an equal basis for the same resource.

Examples of each type of resource can be found in “Hyper-Threading Technology Architecture and Microarchitecture” in this issue of the *Intel Technology Journal*. Consider the operating modes state diagram shown in Figure 1.

---

® Pentium is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.



**Figure 1: Operating Mode State Diagram**

It can be used to illustrate test cases involving the three operating modes and how they affect the four types of resources. At the start of test, both logical processors are reset. After reset, the logical processors vie to become the boot serial processor. Assume logical processor 0 wins and the operating mode is now ST0. All non-duplicated resources are fully devoted to logical processor 0. Next, logical processor 1 is activated and MT-mode is entered. To make the transition from ST0- or ST1-mode to MT-mode, the partitioned structures, which are now fully devoted to only one logical processor, must be drained and divided between the logical processors. In MT-mode, accidental architectural state corruption becomes an issue, especially for the entry-tagged and shared resources. When a logical processor runs the hlt instruction, it is halted, and one of the ST-modes is entered. If logical processor 0 is halted, then the transition is made from MT-mode to ST1-mode. During this transition, the partitioned structures must again be drained and then recombined and fully devoted to logical processor 1. MT-mode can be re-entered if, for example, an interrupt or non-maskable interrupt (NMI) is sent to logical processor 0. The Power Down state is entered whenever the STP\_CLK pin is asserted or if both logical processors are halted.

Now contrast this to a non-Hyper-Threading Technology-capable processor like the Intel Pentium® 4 processor. For the Pentium 4 processor, there are only three states: Reset, Power Down, and Active, and four state transitions to validate. In addition, there is no need to validate the

® Pentium is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

transitioning of partitioned resources from divided to and from combined.

### Architectural State Space

The creation of three operating modes has a material impact on the amount of architectural state space that must be validated. As mentioned earlier, the AV team develops single-threaded tests that fully explore the IA-32 Instruction Set Architecture and architectural space. A single-threaded test has just one thread of execution, meaning that it can run on only one logical processor. A multi-threaded test has two or more threads of execution, meaning that it can run and use two or more logical processors simultaneously.

To validate both ST0-mode and ST1-mode, all AV tests need to be run on both logical processors. A possible solution to validating the micro-architectural state space might be to take all AV tests and simulate all combinations of them in MT-mode. This proved to be impractical and insufficient because one AV test might be much shorter than the other test so a logical processor is halted and an ST-mode is entered before the MT-mode architectural state is achieved. The practical problem is that while simulating all AV tests in one of the ST modes can be done regularly, simulating the cross-product of all AV tests was calculated to take nearly one thousand years [3]!

The solution was to analyze the IA-32 architectural state space for the essential combinations that must be validated in MT-mode.

A three-pronged attack was used to tackle the challenge of Hyper-Threading Technology micro-architectural state space:

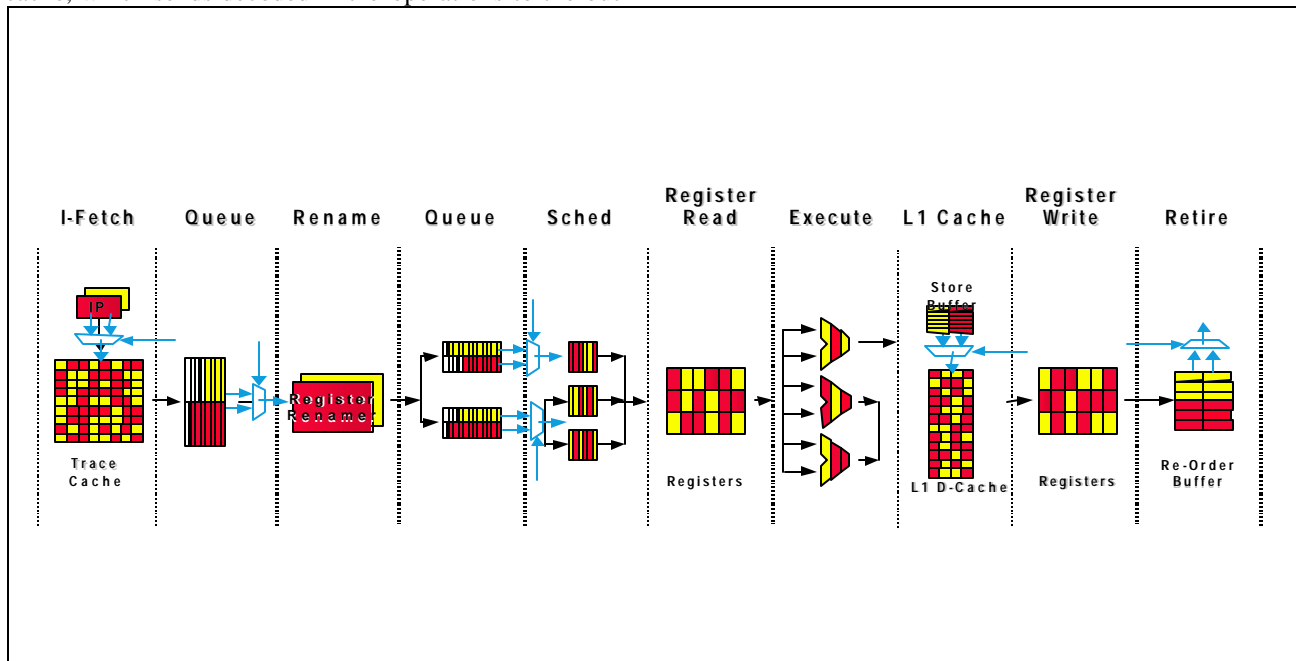
- All AV tests would be run at least once in both ST0- and ST1-mode. This wasn't necessarily a doubling of the required simulation time, since the AV tests are normally run more than once during a project anyway. There was just the additional overhead of tracking which tests had been simulated in both ST modes.
- A tool, Mtmerge, was developed that allowed single-threaded tests to be merged and simulated in MT-mode. Care was taken to adjust code and data spaces to ensure the tests did not modify each other's data and to preserve the original intentions of the single-threaded tests.
- The MTV team created directed-random tests to address the MT-mode architectural space. Among the random variables were the instruction stream types: integer, floating-point, MMX, SSE, SSE2, the instructions within the stream, memory types, exceptions, and random pin events such as INIT,

SMI, STP\_CLK and SLP. The directed variables that were systematically tested against each other included programming modes, paging modes, interrupts, and NMI.

## CONTROLLABILITY

The implementation of Hyper-Threading Technology used multiple logical processor selection points at various pipeline stages. There was no requirement that all selection points picked the same logical processor in unison. The first critical selection point is at the trace cache, which sends decoded micro-operations to the out-

of-order execution engine. This selection point uses an algorithm that considers factors such as trace cache misses and queue full stalls. Hence, controllability can be lost even before reaching the out-of-order execution engine. In addition to the logical processor selection points, controllability is lost because uops from both logical processors are simultaneously active in the pipeline and competing for the same resources. The same test run twice on the same logical processor, but with different tests on the other logical processor used during both simulations, can have vastly different performance characteristics.



**Figure 2: Logical processor selection point**

Figure 2 shows some of the critical logical processor selection points and provides a glimpse into how interacting logical processors can affect their performance characteristics. The independent selection points coupled with the out-of-order, speculative execution, and speculative data nature of the microarchitecture obviously resulted in low controllability at the full-chip level. The solution to the low controllability was the use of the Cluster Test Environment [1] coupled with coverage-based validation at the CTE and full-chip levels.

The Cluster Test Environments allow direct access to the inputs of a cluster that helps alleviate controllability issues, especially in the backend memory and bus clusters. However, logical processor selection points and other complex logic are buried deep within the clusters. This meant that coverage-based validation coupled with directed-random testing was needed to ensure all

interesting boundary conditions had been validated. Naturally, cluster interactions can be validated only at the full-chip level and again coverage-based validation and directed-random testing were used extensively.

## Boundary Conditions

Hyper-Threading Technology created boundary conditions that were difficult to validate and had a large impact on our validation tool suite. Memory ordering validation was made more difficult since data sharing between logical processors could occur entirely within the same processor. Tools that looked only at bus traffic to determine correct memory ordering between logical processors were insufficient. Instead, internal RTL information needed to be conveyed to architectural state checking tools such as Archsim-MP, an internal tool provided by Intel Design Technology.

While ALU bypassing is a common feature, it becomes more risky when uops from different logical processors are executing together. Validation tested that cross-logical-processor ALU forwarding never occurred to avoid corruption of each logical processor's architectural state.

### New Validation Concerns

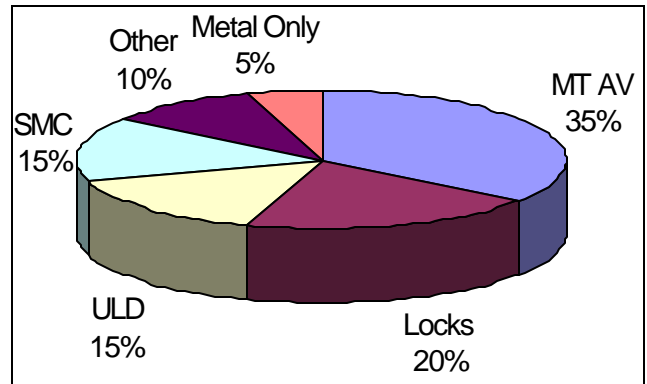
Hyper-Threading Technology adds two new issues that need to be addressed: logical processor starvation and logical processor fairness. Starvation occurs when activity on one logical processor prevents the other from fetching instructions. Similar to starvation are issues of logical processor fairness. Both logical processors may want to use the same shared resource. One logical processor must not be allowed to permanently block the other from using a resource. The validation team had to study and test for all such scenarios.

### BUG ANALYSIS

The first silicon with Hyper-Threading Technology successfully booted multi-processor-capable operating systems and ran applications in MT-mode. The systems ranged from a single physical processor with two logical processors, to four-way systems running eight logical processors. Still, there is always room for improvement in validation. An analysis was done to review the sources of pre-silicon and post-silicon bugs, and to identify areas for improving pre-silicon Hyper-Threading Technology validation.

To conduct the analysis of Hyper-Threading Technology bugs, it was necessary to define what such a bug is. A Hyper-Threading Technology bug is a bug that broke MT-mode functionality. While a seemingly obvious definition, such tests were found to be very good at finding ST-mode bugs. The bugs causing most MT-mode test failures were actually bugs that would break both ST-mode and MT-mode functionality. They just happened to be found first by multi-threaded tests. Every bug found from MT-mode testing was studied to understand if it would also cause ST-mode failures. The bugs of interest for this analysis were those that affected only MT-mode functionality. The bug review revealed the following:

- Eight percent of all pre-silicon SRTL bugs were MT-mode bugs.
- Pre-silicon MT-mode bugs were found in every cluster and microcode.
- Fifteen percent of all post-silicon SRTL bugs were MT-mode bugs.
- Two clusters [2] did not have any MT-mode post-silicon SRTL bugs.



**Figure 3: Breakdown of post-silicon MT-Mode bugs**

Figure 3 categorizes the post-silicon MT-mode bugs into the functionality that they affected [2, 3]. Multi-Threading Architectural Validation (MT AV) bugs occurred where a particular combination of the huge cross product of IA-32 architectural state space did not function properly. Locks are those bugs that broke the functionality of atomic operations in MT-mode. ULD represents bugs involving logical processor forward progress performance degradation. Self-Modifying Code (SMC) bugs were bugs that broke the functionality of self or cross-logical processor modifying code. Other is the category of other tricky micro-architectural boundary conditions. Metal Only is an interesting grouping. We found that post-silicon MT-mode bugs were difficult to fix in metal only steppings and often required full layer tapeouts to fix successfully. Metal Only are the bugs caused by attempting to fix known bugs in Metal Only tapeouts.

### IMPROVING MULTI-THREADING VALIDATION

Clearly, with MT-mode bugs constituting nearly twice the number of post-silicon bugs, 15% versus 8% of the pre-silicon bugs, coupled with the high cost of fixing post-silicon MT bugs (full layer versus metal tapeouts), there is an opportunity for improving pre-silicon validation of future MT-capable processors. Driven by the analysis of pre- and post-silicon MT-mode bugs [2, 3], we are improving pre-silicon validation by doing the following:

- Enhancing the Cluster Test Environments to improve MT-mode functionality checking.
- Increasing the focus on microarchitecture validation of multi-cluster protocols such as SMC, atomic operations, and forward progress mechanisms.
- Increasing the use of coverage-based validation techniques to address hardware/microcode interactions in the MT AV validation space.

- Increasing the use of coverage-based validation techniques at the full-chip level to track resource utilization.

Done mainly in the spirit of continuous improvement, enhancing the CTEs to more completely model adjacent clusters and improve checking will increase the controllability benefits of CTE testing and improve both ST- and MT-mode validation. Much of the full-chip microarchitecture validation (uAV) had focused on testing of cluster boundaries to complement CTE testing. While this continues, additional resources have been allocated to the multi-cluster protocols mentioned previously.

The MTV team is, for the first time, using coverage-based validation to track architectural state coverage. For example, the plan is to go beyond testing of interrupts on both logical processors by skewing a window of interrupt occurrence on both logical processors at the full-chip level. In addition, this will guarantee that both logical processors are simultaneously in a given architectural state.

The MTV team is also increasing its use of coverage to track resource consumption. One case would be the filling of a fully shared structure, by one logical processor, that the other logical processor needs to use. The goal is to use coverage to ensure that the desired traffic patterns have been created.

Nevertheless, these changes represent fine-tuning of the original strategy developed for Hyper-Threading Technology validation. The use of CTEs proved essential for overcoming decreased controllability, and the division of MT-mode validation work among the existing functional validation teams proved an effective and efficient way of tackling this large challenge. The targeted microarchitecture boundary conditions, resource structures, and areas identified as new validation concerns were all highly functional at initial tapeout. Many of the bugs that escaped pre-silicon validation could have been caught with existing pre-silicon tests if those tests could have been run for hundreds of millions of clock cycles or involved unintended consequences from rare interactions between protocols.

## CONCLUSION

The first Intel® microprocessor with Hyper-Threading Technology was highly functional on A-0 silicon. The initial pre-silicon validation strategy using the trinity of coverage-based validation, CTE testing, and sharing the

validation work was successful in overcoming the complexities and new challenges posed by this technology. Driven by bug data, refinements of the original validation process will help ensure that Intel Corporation can successfully deploy new processors with Hyper-Threading Technology and reap the benefits of improved performance at lower die size and power cost.

## ACKNOWLEDGMENTS

The work described in this paper is due to the efforts of many people over an extended time, all of whom deserve credit for the successful validation of Hyper-Threading Technology.

## REFERENCES

- [1] Bentley, B. and Gray, R., "Validating The Intel Pentium 4 Processor," *Intel Technology Journal Q1, 2001* at [http://developer.intel.com/technology/itj/q12001/article/art\\_3.htm](http://developer.intel.com/technology/itj/q12001/article/art_3.htm)
- [2] Burns, D., "Pre-Silicon Validation of the Pentium 4's SMT Capabilities," *Intel Design and Test Technology Conference, 2001*, Intel internal document.
- [3] Burns, D., "MT Pre-Silicon Validation," *IAG Winter 2001 Validation Summit*, Intel internal document.

## AUTHOR'S BIOGRAPHY

**David Burns** is the Pre-Silicon Hyper-Threading Technology Validation Manager for the DPG CPU Design organization in Oregon. He has more than 10 years of experience with microprocessors, including processor design, validation, and testing in both pre- and post-silicon environments. He has a B.S. degree in Electrical Engineering from Northeastern University. His e-mail address is [david.w.burns@intel.com](mailto:david.w.burns@intel.com)

Copyright © Intel Corporation 2002. Other names and brands may be claimed as the property of others.

This publication was downloaded from <http://developer.intel.com/>

Legal notices at <http://developer.intel.com/sites/corporate/tradmarx.htm>

---

®Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.