**Alpha 21164 Manufacturing Test Development and Coverage Analysis**

**CAROL J. STOLICHNY**
**RICHARD DAVIES**
**PAMELA MCKERNAN**
**TUYN TRUONG**
Compaq Computer Corporation

**Today’s competitive market** demands high quality levels for electronic devices. This is especially true for state-of-the-art microprocessors. Unfortunately, the full-custom design methodology and performance constraints of the Alpha 21164, a deep submicron, superscalar microprocessor, precluded the brute force use of design-for-test (DFT) techniques such as full-scan insertion and automated test pattern generation. Further, scan testing alone cannot guarantee the desired quality level for microprocessors in this genre. As a result, functional tests play a key role in achieving the Alpha 21164’s electrical quality goals.

Generating a comprehensive test set for a high-performance microprocessor without the benefit of automated tools is an enormous challenge. Designers must craft many tests by hand, and coverage analysis of the resultant vectors can be costly and time-consuming (even with the simplest of fault models).

For the Alpha 21164, a proactive effort began early in the design cycle to meet this challenge. This effort yielded an initial functional vector set leveraged from a variety of test sources. We used a cost-effective, distributed software simulation scheme to grade the faults of the vectors and guide test improvements. Our team identified several structural blocks commonly found to have coverage shortfalls. We also adopted new heuristics to prioritize additional test development. Our final analysis used manufacturing data to determine the effectiveness of the different methods and to understand the role that functional tests play in identifying defective devices.

**Design attributes and test methodology**

The Alpha 21164 is a superscalar, quad-issue microprocessor operating at frequencies up to 625 MHz. Currently in volume production, this second-generation implementation of the Alpha architecture was introduced in 1995. Table 1 summarizes key product and process features.

With roughly 80% of the microprocessor’s transistors dedicated to on-chip cache memory, the remaining transistors and datapath consume a majority of the die area. We focused our test development effort on this remaining area while testing the caches using industry standard methods with DFT assistance. The Alpha 21164’s DFT features include built-in self-test and self-repair of the instruction cache along with observable linear feedback shift registers for fault coverage improvement. In addition, architectural test features overrode normal chip functions or enabled operating modes that increased controllability and observability (such as embedded memory force-hit modes and arbitration-squelching capabilities).

We combined custom DFT features with a robust manufacturing test strategy to achieve the Alpha 21164’s test quality goals. Automated testing includes a rigorous combination of functional and parametric test conditions. For example, we ran all functional tests at the full device-under-test op-
erating speed to test the core logic in worst-case conditions for timing and delay faults.

**Initial vector set development**

The initial manufacturing test vector development heavily leveraged the design verification process and derived tests from a range of existing and new sources to provide reasonable baseline test coverage. We used a variety of test sources.

**Architectural verification tests.** Architectural verification tests provided a ready source of test stimulus for the 21164 macroarchitecture. These tests ensured that a specific design complied with the instruction set architecture specifications. Since these tests are usually developed and shared across multiple product generations, this test source is rich and exhaustive.

**Pseudorandom test scripts.** Pseudorandom design verification exercisers offer a quick means of touching implementation-specific logic functions with modest engineering effort and were useful test pattern sources for device prototyping. To augment the broadside benefit of pseudorandom tests, we used coverage analysis tools to identify and capture random test sequences that exercised unique logic functions or complex timing interactions.3

**Focused design verification tests.** Focused tests targeted features specific to the Alpha 21164 microarchitecture. Typically hand-generated, these implementation-specific tests provided the most exhaustive source of manufacturing vectors, including tests for worst-case conditions.

**System events and demons.** The design verification tool suite featured a multipurpose system interface model, or transactor, that mimics the system configuration options that Alpha 21164 supports. It also invokes spurious system events, or demons.4 Using programmable option selection, our test development environment automated multiple test pattern generation from a single test sequence.

**Structural coverage tests.** Lastly, manufacturing-specific structural tests augmented the traditional design verification test suite. We generated these tests to exercise both random and arrayed logic other test sources or BIST features do not cover. We developed test plans for each partition and used toggle coverage analysis to guide structural test development.

**Evaluating vector set coverage**

Once we developed the initial vector set, we selected the classical single stuck-at-fault model to evaluate its coverage. While the single stuck-at fault does not model all CMOS defects, its simplicity makes it a practical candidate for measuring completeness of test patterns, as well as for providing a mechanism to guide test development.

We considered two options for fault simulation: the cycle-based logic simulator used for design verification and a commercially available hardware accelerator. Our cycle-based simulator can simulate stuck-at faults, and it includes a set of tools to manage and automate the fault simulation. The hardware accelerator offers the possibility of increased performance over the software solution. However, a hardware accelerator would have required a significant capital expenditure due to the size and complexity of the design and a nontrivial effort to convert the model to its required format.

Additionally, we noted that recent experiments in simulating faults in the first Alpha implementation (21064) with a hardware accelerator had not demonstrated a clear price/performance advantage. Therefore, we graded the test vectors using the proprietary software simulator. We dedicated a large cluster of Alpha workstations to this work at a fraction of the hardware accelerator’s cost. Figure 1 illustrates the basic process flow for the manufacturing test coverage analysis of the Alpha 21164.

**Table 1. Alpha 21164 device features and characteristics.**

<table>
<thead>
<tr>
<th>Features</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistor count</td>
<td>9.6 million</td>
</tr>
<tr>
<td>Die cavity size</td>
<td>14.6 mm x 14.66 mm</td>
</tr>
<tr>
<td>Process parameters</td>
<td>2.0-V, 0.35-micron CMOS</td>
</tr>
<tr>
<td>Clock speeds</td>
<td>366 MHz to 625 MHz</td>
</tr>
<tr>
<td>On-chip caches</td>
<td>8-Kbyte L1 instruction cache</td>
</tr>
<tr>
<td></td>
<td>8-Kbyte L1 data cache</td>
</tr>
<tr>
<td></td>
<td>96-Kbyte L2 data/instruction cache</td>
</tr>
<tr>
<td>Package</td>
<td>499-pin IPGA</td>
</tr>
</tbody>
</table>

**Figure 1. Basic flow for test grading.**
Test partitioning. For fault simulation purposes, we divided both the logic and the functional test suite into partitions. This partitioning reduced the simulation time required because only tests that target a specific functional section were run on the faults within that section. A standard test-naming convention easily identified the primary partition associated with each test. Toggle coverage data facilitated both the partitioning and ordering of tests to optimize simulation performance.

Pattern conversion. The tester-ready pattern is converted to input stimulus and output comparison. The custom conversion tool uses the test program timing database to accurately reflect conditions seen by the device under test on the tester. For example, the input file preserves setup and hold timings, and the output comparison file reflects any output masking done on the tester. The output file is a snapshot of known-good circuit behavior.

Fault simulator development. Our proprietary compiled-code simulator provided a gate-level model offering runtime insertion of single stuck-at faults. While the design process provides the most substantial pieces of the Alpha 21164 model, we needed to remove certain design optimizations to achieve an appropriate abstraction level. With the required modifications, the resultant model of the Alpha 21164 is an efficient fault simulator.

A key feature of the simulator is its ability to run up to 64 different simulations in parallel on a single CPU. The Alpha architecture’s 64-bit word represents network nodes, with each bit representing that node’s binary logic value in a unique simulation. The 64 faulty circuits can run in slightly more time than one serial simulation, subject to Amdahl’s Law, with nonparallel tasks limiting performance.

The output comparison pattern file provides the fault-detection mechanism for each simulation. The simulator compared data from the output file with the faulty model pin state at each cycle. In the event of a mismatch, the test detects a fault and suspends that simulation. The fault is then logged, and a new fault is inserted in its place. In addition to the primary pin output reference, the inputs to the observable LFSR chains provide additional fault simulation comparison points. We saved simulation time by dropping faults when they propagated to a linear feedback shift register input rather than waiting for the signature scan-out at the test end.

Fault universe and sample generation. For each partition, we generated a fault library from the gate-level net list. We then eliminated untestable faults, such as nodes with no fan-out, from the fault population. We did not include the caches since the RAM test algorithms employed achieve 100% single stuck-at-fault coverage. From the resultant fault universe, we then selected a random and homogeneous sample to achieve the desired confidence interval for the measured result. For the sampling, we assumed all faults to have equal failure probability; each partition is represented proportionally.

Distributed fault simulation. We performed the fault simulation on a large cluster of Alpha workstations dedicated solely to this purpose. We also developed in-house tools to support the fault simulation effort. These tools handled the tasks of creating simulation batch jobs, managing the fault library, job distribution and monitoring on workstations, and recovering from hardware or software problems without user intervention or loss of simulation time. Figure 2 depicts the distributed fault simulation methodology.

Coverage improvement/enhancement

The results from the initial fault grading guided the first iteration test enhancement efforts. We prioritized investigation of undetected faults in several ways, with the overall goal to achieve uniformly high coverage across the die area.

Within that constraint, we made clusters of undetected faults the initial test development focus. In most cases, we wrote tests to cover all faults in a particular logic section, not just the faults from the sample. After filling large coverage holes, we ranked the remaining undetected faults by the node’s total interconnect capacitance. Interconnect capacitance is an indicator of the critical area of a node and may be linked to its probability of failure due to process de-
We selected this prioritization scheme in an effort to increase coverage of real defects.

Our analysis uncovered several circuits typically found to have poor coverage with the initial vector set. Since we derived the suite largely from design verification tests, we called these faults DVT hard faults. Often, several instances of DVT hard faults result in a systemic coverage shortfall. The common problem circuits include:

- content-addressable memory arrays
- bypass datapaths
- register conflict logic
- register file and internal processor registers

Note that each of these circuits was exercised to the criteria required for functional design verification. However, random testing may have achieved exhaustive logic verification of these circuits, which would not have yielded an efficient test set.

**Content-addressable memory arrays.** These circuits accounted for the predominant coverage shortfall in the design due to their widespread use. For example, designers used the basic structure in both the instruction and data address translation buffers and in read, write, and cache victim buffers.

The wired-OR, pulldown match line (Match0 and Match1 in Figure 3) requires an exhaustive bit-by-bit test to excite the stuck-at-zero faults on nodes NZ and NZC. We can observe this fault when the bit under test was supposed to mismatch but didn’t due to the fault’s presence.

In addition to the bit-match lines, the initial test set did not exhaustively test all of the data storage elements in the content addressable memory. Tests written to fully cover these arrays contributed approximately 4% to the overall stuck-at fault coverage for the first iteration improvement activity.

**Bypass datapaths.** The bypass datapaths comprised the only coverage shortfall in the integer execution unit. These paths connect to and from the adders, comparison and logic units, and the load/store data bus. We discovered that our initial test set didn’t excite all the possible data forwarding conditions and states. In addition, the manufacturing test programmer needed to give extra consideration to making the effects of stuck-at faults in a bypass path observable for certain comparison operations.

**Register conflict logic.** The register conflict logic was another DVT hard fault type. Similar to the data bypasses, this coverage shortfall was due to a general lack of register conflict conditions, and a failure to construct all conflict combinations in the focused test suite.

**Register file and internal processor registers.** The verification tests used for test patterns did not exercise registers that were not critical to the chip’s basic functionality in an application (such as the performance event counter). We also found some registers or specific bit fields difficult to control and observe because they could only be read or written by hardware functions. Figure 4 illustrates each coverage shortfall mechanism’s contribution to the overall single stuck-at-fault coverage improvement that we discovered in the first iteration.

In addition to the test development guided by fault sim-

**Figure 3.** Data translation buffer content-addressable memory match logic circuit.

**Figure 4.** First iteration coverage improvement categories.
ulation data, system-level diagnostics served as a coverage improvement test source. We converted instruction sequences from system-level test that uniquely identify defective devices to manufacturing test patterns. Incremental grading of these faults sometimes, although not always, increased the measured coverage. We were not surprised by the lack of an increase in fault coverage—we attributed this to sampling effects or potential physical defect coverage not classified as stuck-at faults.6

Results

We have identified and described in detail several coverage shortfalls. For each case, we added tests to the manufacturing vector suite. Since test development is an iterative process, we selected another fault sample and remeasured the coverage. This verification ensured an overall improvement (not just for a specific sample of faults), and provided additional data to drive subsequent test coverage improvement iterations. Figure 5 shows results from the first iteration of test coverage improvement activities, including the 3-σ statistical confidence interval (the I on top of each bar) associated with each sample.

These results reflect a snapshot of the iterative process; they do not necessarily reflect current or future fault coverage status. Also, the test partitioning methodology may yield a conservative coverage metric. Nonetheless, the results demonstrate that the initial test pattern generation strategy yielded a vector set with relatively high stuck-at-fault coverage. Moreover, the initial fault coverage improved more than 6% as a result of an additional test development effort that brought all partitions to a similar coverage level.

We should note that the initial coverage of the third partition was high relative to other partitions, and we did not develop new tests for that section during the first coverage improvement iteration. Consequently, we expected a variation in the statistical fault coverage of the second sample, and the lower coverage fell within the computed confidence interval.

The Alpha 21164 observable linear DFT feature enhanced the manufacturing pattern development activity in several ways. Foremost, the initial test pattern suite coverage increased 2.5% due to the added observability. This coverage boost reduced not only the test development efforts but also the fault simulation requirements by detecting faults earlier than the test could detect them at the primary outputs. Test enhancements could also take advantage of this feature by propagating fault effects to local observability linear feedback shift register chains, thus creating more efficient tests.

**Effectiveness of test development methods.** We used several test sources to achieve the stuck-at-fault coverage measured in the first iterative fault simulation. Although we chose not to present detailed data, we want to comment on the overall value and effectiveness of the various test sources.

For three of the functional partitions, a single architectural verification test or a generic pseudorandom test sequence reached minimum 75% single stuck-at-fault coverage. Since these tests are essentially “free” (meaning no incremental test development cost) and provide such quick and broad test coverage, we can’t underestimate their value for manufacturing test. However, limited diagnostic capability is one significant random test drawback.

Focused functional tests developed specifically for design verification, toggle coverage, or single stuck-at-fault coverage are essential for increasing the fault coverage to a very high level. They are also useful for design debug and failure isolation. Naturally, we included the test methods required to cover the identified DVT hard faults in the initial test development activity efforts for future generations. Although focused tests are typically handcrafted, we should note that many such tests are algorithmic in nature, and are candidates for built-in testability or automated test generation.

Difficult-to-excite faults, such as complicated event interactions or timing sequences, represent the smallest coverage improvement opportunity, yet they require the largest test development effort. In absence of structured DFT, we can best cover these faults by applying coverage analysis tools to pseudorandom test generation. We did not fully exploit this method in our work—it presents an opportunity for future development.

To rate the effectiveness of the nodal capacitance prioritization metric, we collected and analyzed per-pattern fail data for each Alpha 21164 device from the online product.
The test program flow facilitated unique identification of devices that failed one or more of the new patterns but none of the existing patterns. Figure 6 illustrates the defective device capture rate as well as the single stuck-at-fault, and the total interconnect capacitance coverage increase for several new tests. The data illustrates a positive correlation between interconnect capacitance coverage and the defective device capture rate, while the single stuck-at-fault coverage correlation is unclear.

The illustration in Figure 6 shows that the use of nodal capacitance as a prioritization heuristic proved successful in achieving the goal of improving overall device quality in an efficient manner. However, it is not known whether the observed trend is actually due to an increased probability of failure or the peripheral coverage effects of global signals.

Use of stuck-at-fault model. While controversy exists over the use of the stuck-at-fault model as a defect coverage metric, it is the only model that could realistically be used for a large-scale design like the Alpha 21164. We primarily used this fault coverage metric to identify coverage shortfalls and as a relative, not absolute, measure of quality. Nonetheless, stuck-at-fault coverage can reasonably estimate defect coverage when applied to functional tests due to its ability to cover nontargeted defects. This is particularly true for microprocessors designed to push the performance and process limits of the technology. At-speed functional testing sensitizes coupling, noise, data dependency, thermal, and speed conditions that existing ASIC scan test methods might not adequately cover.

To illustrate the effectiveness of functional tests at detecting both target and nontarget faults for the Alpha 21164 microprocessor, we again analyzed manufacturing data. Functional tests run at several operating conditions including nominal frequency and voltage, high and low voltage margins, and full design-under-test speed. We can identify manufacturing defects that behave as stuck-at-faults at nominal conditions (as well as some nontarget faults). Further, we assert that devices that pass all coverage improvement patterns at nominal conditions, but are rejected at a voltage or speed tolerance, represent the nontarget defect classes. Figure 7 shows the breakdown of capture category for one subset of added patterns.

The data illustrates that functional tests capture nearly as many defects not classified as stuck-at-faults as those that are. This trend characterizes not just our new tests but any functional test grouping.

High-performance processor design teams face a difficult task when they attempt to create high-quality manufacturing test vectors in a timely and cost-effective manner. We met this challenge for the Alpha 21164 through a practical blend of techniques to develop a quality initial test vector suite, perform an efficient fault simulation, and enhance the manufacturing test coverage using an iterative process. Moreover, we accomplished this effort with a modest investment in both capital and personnel resources. The manpower resources required to complete the work we've described, including tool and test development and fault simulation represent less than 2% of the total design effort.

In addition, we made several key findings as a result of this work:

- DVT hard faults resulted in systemic coverage shortfalls.
- Nodal capacitance is an effective metric for prioritizing test development.
- Functional patterns provide significant coverage for defects not modeled as single stuck-at-faults when used in typical microprocessor test conditions.

**Figure 6.** Per test fault data.

**Figure 7.** Defective device categories.

<table>
<thead>
<tr>
<th>Test</th>
<th>Defective device capture</th>
<th>Number of faults detected</th>
<th>Interconnect capacitance of faults detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Alpha 21164 manufacturing pattern development effort led to the creation of tests that other Alpha designers have successfully leveraged for initial test suite development. We also identified several opportunities to improve tools and methods as a result of this activity. Further, through this work, we learned that we could have increased the productivity of the test development activity by automated identification and removal of certain types of undetectable faults and by improving fault simulator performance. Nonetheless, the manufacturing functional test pattern development methods presented here, in combination with the custom DFT and manufacturing test strategies employed, meet the device quality and cost goals of the Alpha 21164 microprocessor.

Acknowledgments

We extend our sincere gratitude to the following people for their numerous contributions and consistent support: Dilip Bhavsar, Dave George, Bill Grundmann, Michael Kantrowitz, John Kapp, Heidi Kvinge, Bill LaPrade, Suresh Nadig, Dave Shepard, Alexander Stein, Rishan Tan, Scott Taylor, and Xiaocun Xu.

References


Carol J. Stolicny is a principal engineer in the Alpha CAD and Test Group at Compaq Computer Corp. in Shrewsbury, Massachusetts. She is currently developing test strategies and solutions for future generations of Alpha microprocessors. Her research interests include microprocessor test and verification, fault simulation and modeling, and failure diagnosis. Stolicny received her BS in electrical engineering from the University of Illinois. She is a member of the IEEE and the Computer Society.

Richard Davies is a principal engineer in the Alpha CAD and Test Group at Compaq. His responsibilities have included design verification, silicon debug, fault-grading, manufacturing test development, and pattern generation. Davies received his BSEE from the University of Cincinnati and his MSEE from Northeastern University. He is a member of the IEEE and the Computer Society.

Pamela McKernan is a hardware engineer in the Alpha CAD and Test Group at Compaq. Since joining the group, she has worked on manufacturing test solutions for two Alpha microprocessors as well as the StrongArm microprocessor. McKernan received her BS in computer engineering from Pennsylvania State University.

Tuyen Truong is a hardware engineer in the Alpha CAD and Test Group at Compaq. She is currently involved in the design and support of test data analysis tools for the Alpha Product and Test Engineering Group. Tuyen received her BS in computer engineering from Pennsylvania State University.

Direct questions concerning this article to Carol J. Stolicny, Compaq Computer Corp., 334 South Street, Shrewsbury, Massachusetts 01545; carol.stolicny@digital.com.