Data Center Silent Data Errors

Technical Paper – Volume 3

March 2024

Revision: 1.1

Document Number: 788204

Contents

Figures

Revision History

Revision Number	Description	Date
1.0	Initial release of the document.	September 2023
1.1	Added volume 3 terminology.	March 2024

1 Document Overview

Data errors are not new to the computing industry. Alpha particle-induced soft errors in memory and magnetic hard drive errors have long been well researched and documented. More recently, technical papers about Silent Data Errors (SDEs) caused by a defect within the microprocessor have been published by Intel[1], Google[2], and Meta[3], and the topic has been discussed at industry conferences[4].

Computer systems typically have a variety of ways to detect errors. For example, data paths and memory storage may include parity or error check and correction (ECC) schemes; and network packets have error-detecting or error-correcting algorithms such as cyclic redundancy checks (CRC). In all these cases, system hardware detects the error and take action to prevent the incorrect data from being used by an application. In rare cases, an operation takes place in the processor where the result is incorrect. An event like this is considered a silent data error if such calculations do not have an error checking mechanism.

Silent data errors are an industry challenge and continue to be rare events[2]. But in a data center running multiple millions of servers, 24 hours a day, the rare event becomes an expected occurrence, and the system administrator must be proactive in managing the event.

Intel is leading the industry in developing and delivering silicon and software tools to maximize the data integrity of computer systems and end user applications.

In anticipation that customers would want tools to detect SDE in their environment, Intel delivered the Intel® Data Center Diagnostics Tool (DCDiag) for use in manufacturing and fleet maintenance. To facilitate continued innovation, Intel initiated the Open Data Center Diagnostics Project for the open-source community.

A discussion of SDE, Intel's test philosophy, and the Intel Data Center Diagnostics Tool is the focus of this paper.

Keywords: SDE, SDC, Silent Data Corruption, Silent Data Errors, Data Center Diagnostics, DCDiag.

[1] David P. Lerner; Benson Inkley; Shubhada H. Sahasrabudhe; Ethan Hansen; Luis D. Rojas, Van Der Ven, Arjan; Optimization of Tests for Managing Silicon Defects in Data Centers https://ieeexplore.ieee.org/document/9983919

[2] P. Hochschild, R Govindaraju, D. Culler, P. Turner, P. Ranganathan, A. Vahdat, J. Mogul, "Cores that don't count," HotOS '21, May 31-June 2, 2021, Ann Arbor, Michigan, USA https://doi.org/10.1145/3458336.3465297

[3] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah, S. Sankar, "Silent Data Corruptions at Scale," arXiv:2102.11245v1 (https://arxiv.org/abs/2102.11245)

[4] PANEL OCP HW Operation at Scale Reliability to Address Silent Data Corruptions, November 9, 2021 https://www.youtube.com/watch?v=3yhg4Gt8M_E

2 Introduction

2.1 What Is a Silent Data Error?

Many terms have been used to describe the phenomenon when a processor produces an undetected (or silent) incorrect result. In its papers, Meta called this silent data corruption and Google referred to it as corrupt execution error. Intel uses the term Silent Data Error (SDE). SDE can be the result of an external influence, such as an alpha particle changing the state of a data bit in DRAM or cache memory. It can also occur when the processor incorrectly performs an operation (multiply, divide, move, rotate, and so forth) that produces the wrong result. SDEs are not unique to microprocessors or specific vendors. Any graphics chip, network controller, AI processor, or other silicon device has the potential to generate an incorrect result.

Silent data errors are of concern to customers because they can have a variety of consequences, ranging from very minor – a pixel in a photograph may be the wrong color, to more serious – a mathematical calculation may be incorrect, or an encryption key may be mistranslated such that encrypted data cannot be decrypted.

3 SDE Causes and Rate of Occurrence

3.1 SDE Causes

SDEs can be caused by random defects introduced into the device during the manufacturing process. In many cases, these defects are severe enough that they cause problems in the processor that are easily detected and rejected during manufacturing test. Other times, the defects are subtle and create circuit marginalities that fail only under the right combination of temperature, voltage, frequency and instruction sequence or data set. Detecting these marginal failures during testing can be extremely difficult because it is impossible to check every combination of conditions and potential workloads.

Defects can also be latent, meaning they do not show up until after the processors have been running for some time. In the example cited by Meta, core 59 on one processor consistently returned a result of 0 when calculating Int(1.1^{53}), rather than 156. But the same core would return the correct value of 142 for Int(1.1^{52}). All other cores on the die, and all other processors in the system, worked correctly. It is unknown if this defect was present when the part left the factory, or if the circuit in question failed sometime after it was put into service. (Specific processor or manufacturer was not identified.)

Circuit marginalities can result in random, or unrepeatable, SDEs. As mentioned above, a marginal circuit is one that usually operates correctly, but under the right conditions of temperature, voltage, frequency, data set, instruction sequence, and so forth, does not. Reproducing a marginality induced SDE can be extremely difficult because a slight change in one, or more, of the operating parameters may make the difference between correct and incorrect circuit behavior. Operating conditions, design errors, manufacturing defects and silicon aging can all cause a circuit to be marginal.

3.2 How Often Do Silent Data Errors Occur?

Defect rates are described in terms of Defects Per Million (DPM) which states how many parts in a population of one million devices might contain a defect. Given that SDEs are silent, it is difficult to measure accurately how frequently they occur. Evaluating data from Intel experiments and customer reported errors, Intel estimates that SDEs are a small portion of the expected DPM that a customer may experience.

What this means for customers is that they may never experience SDEs on a device that has a marginal defect. If the customer's system implementation (meaning power delivery, cooling, bus timing, signal quality, and so forth) and software application do not create the conditions necessary to make the marginality manifest as an SDE, they will not experience SDEs with that device. Using that same part in a different system with different application software may create the conditions necessary for the marginality to manifest as an SDE. In that case, the customer may experience SDE events. When it comes to circuit marginalities, predicting the frequency of failures is very difficult.

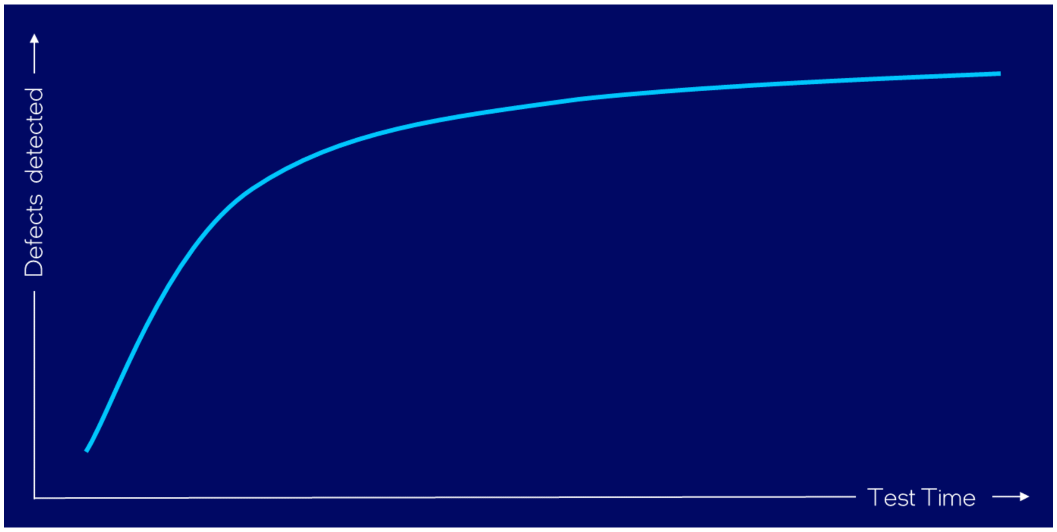
4 Intel® Processor Testing Methodology

4.1 Overview

Every Intel processor runs through an extensive testing sequence which is designed to efficiently detect as many defects as possible. After packaging and burn-in, each part is checked on an Automated Test Equipment (ATE) system that varies temperature, voltage and frequency while running various tests. Test times at this stage are usually in the range of a few minutes. ATE provides high fault coverage, but it does not test the processor in a system environment (hard drive, memory, network card, and so forth), which is how customers will use it. After ATE, each device is installed in a system environment where tests using multiple operating system environments are run.

A distinction should be made between the initial outgoing SDE DPM and what end customers will observe in their data centers. Since it can take many minutes of system level testing to achieve a very low rate of SDE DPM, that indicates the defects in the outgoing parts are probably of a marginal, or intermittent, nature. In other words, after extensive testing, some parts have not seen the right conditions to allow their defects to be detected. The following figure displays a typical defect-detected vs time plot for system level testing.

Figure 4-1. Conceptual view of processor test time vs. defects detected. Longer test times find more defects, but at a diminishing rate of return.



Intel has developed several test programs which are used on Intel® Xeon® processors during manufacturing. These programs check every instruction on each core, all the caches, core-to-core communications, memory interfaces, uncore functions, and so forth, with the goal of exercising a high percentage of transistors in the package. However, it is not possible to test every transistor in the system environment. For example, there is no way to use X86 instructions to purposely create a parity error on an internal bus that would be detected and flagged by the processor. Therefore, the circuits that detect and report internal parity errors cannot be tested at the system level. ATE is required to test such functionality.

4.2 Detecting SDEs with the Intel Data Center Diagnostics Tool (DCDiag)

Intel developed the Intel Data Center Diagnostics Tool (DCDiag) to provide customers with an application to identify faulty processors.

DCDiag is a test tool that integrates some of Intel's system test programs into an easy-to-use package that runs under Linux* or Microsoft* Windows*. A single command initiates and runs DCDiag, which takes about 45 minutes to complete. At the conclusion of the test, DCDiag provides a simple report that summarizes the system configuration and gives a clear indication of pass or fail. Automated scripting can be implemented by the customer to parse the output. If a failure is detected, DCDiag includes an error code that can be interpreted by Intel to help understand the nature of the failure.

DCDiag can be used in a variety of ways depending on the user's goals and requirements. System manufacturers can use DCDiag to confirm that the system is assembled properly, and that the processor is operating correctly. System administrators may use DCDiag as part of their incoming acceptance and burn-in of new systems, and for periodic maintenance of their installed base.

4.3 Error Detection Methodology

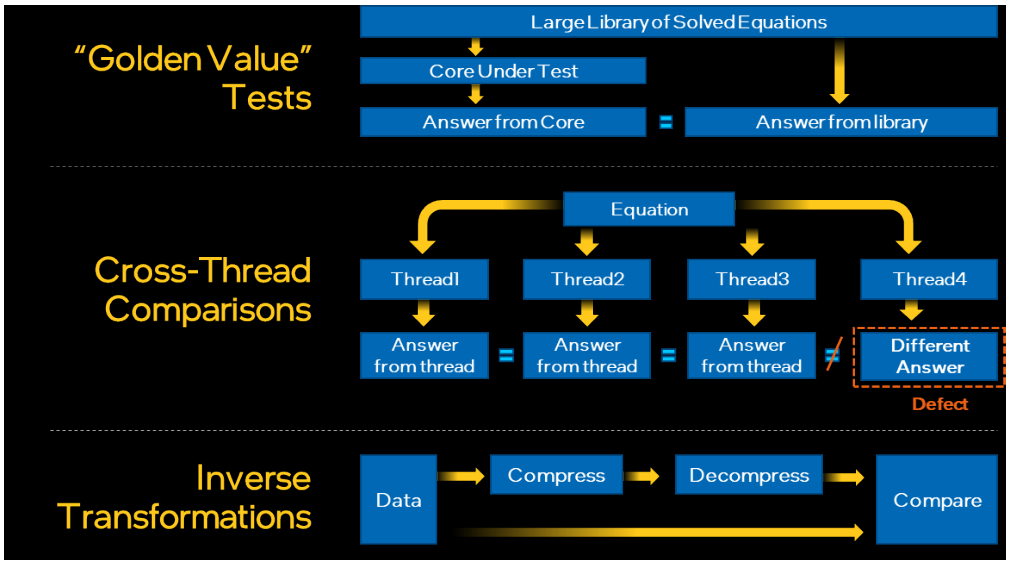
DCDiag uses multiple mechanisms to detect silent data errors. In general, the concept is to perform a calculation or operation repeatedly and then check to confirm that the results are correct; see Figure 4-2. Three examples of software methods to detect processor defects.

Calculating the square root of 2 or the SHA-1 checksum of a fixed input are examples of golden value tests, in which a known result is expected from a specific sequence of operations. In this case, each core in the processor would calculate the same expression and compare it to the expected answer. If the result differs from that even once, the processor is flagged as defective and should be replaced.

In cross-thread comparisons, all cores within the package run the same sequence of instructions using the same dataset. Variability is added to this type of test by using randomly generated datasets and/or randomizing the order or selection of processor instructions. Examples of this are similar to the above case of square root and SHA-1 calculation, but instead of using fixed input and known output, the output generated by each core is compared against that produced by other cores. In simpler terms, an operation such as $D = A + B \times C$ could be used with random values for A, B and C. The value of D is not known at the beginning of the operation since the values of A, B and C are random, but if all cores compute the same result, the processor would be operating correctly. If any core ended up with a different value for D, the processor would be considered defective. Running multiple iterations of a test with randomization of the variables and instruction sequences exercises transistors in more ways than using predefined values and therefore provides higher overall test coverage.

Inverse transformation tests are tests that execute two operations back-to-back and are supposed to arrive at the original input. Examples of this are compression and decompression, or encryption and decryption on a randomly generated dataset. Comparing the original data to the final data confirms if the operation was completed successfully. Any mismatches would indicate a defect in the processor.

Figure 4-2. Three examples of software methods to detect processor defects.



These are just a few examples of how DCDiag searches for silent data errors, but there are other mechanisms that can be used. Additional tests check core-to-core and socket-to-socket communications, caches, and other non-compute functions of the processor.

Some defects may cause non-silent errors, such as a parity error or an invalid operation. DCDiag can find these types of defects too. It all depends on where a defect is located and how it manifests when an operation attempts to use the affected transistors.

5 Summary: Addressing an Industry Challenge

Cloud service providers run millions of processors, 24 hours per day, in data centers around the world. With such large numbers of processors, extremely rare events, such as silent data errors, may occur frequently enough that mitigation is essential. Technical papers and industry conferences have discussed this industry-wide challenge, and this has created an opportunity for the development of tools to find defects that generate SDEs.

Intel is leading the industry in addressing SDE detection by releasing a publicly available software tool called DCDiag. It is available on Intel.com[5] and is free for customers to use on their Intel Xeon processors. Periodic testing of the fleet can find defects that occur over time and reduce the potential for silent data errors.

Additionally, we encourage any researcher interested in working with the open-source community on improving the methods for addressing SDE to contact the OCP workgroup.

Intel recognizes there are many organizations across the industry that are researching ways to identify processor errors in a more effective and efficient manner. Intel's Open Data Center Diagnostics Project (Open DCDiag) is designed to encourage industry test development collaboration. Open DCDiag is a consistent test development framework that invites the creativity of the Open-Source community to enhance cloud fleet management through the development of unique test screens and other innovative solutions. ODCDiag is another way Intel continuously works with the industry to improve Intel Xeon processor-based platform reliability.

[5] https://www.intel.com/content/www/us/en/support/articles/000058107/processors/intel-xeon-processors.html