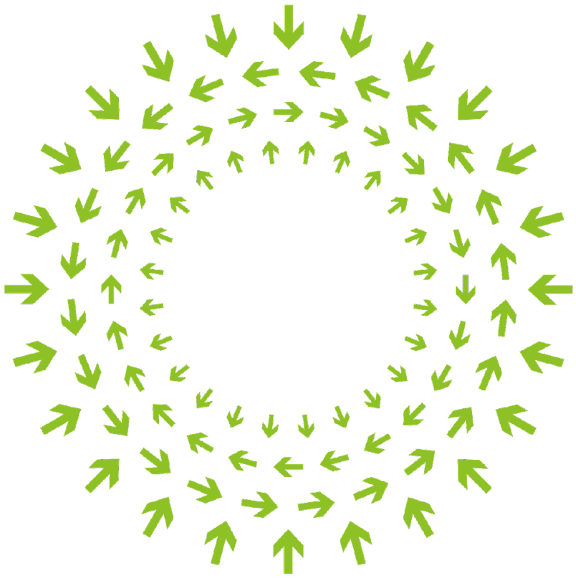

OPEN Compute Project

Server Component Resilience

Revision 0.3
June 30, 2023

Contributors:

Bharath Parthasarathy and Amber Huffman, Google
Harish Dattatraya Dixit and Tejasvi Chakravarthy, Meta Platforms Inc
Rob Chappell, Microsoft
Vilas Sridharan and Sankar Gurumurthy, AMD
Thiago Macieira, Intel
Reiley Jeyapaul and Lisa Minwell, ARM
Lidia Warnes, NVIDIA

Table of Contents

Glossary of Terms

This section provides the glossary used in this specification. Note that it is not organized in alphabetical order but in sequential order to best understand the terms and definitions as they flow through the document.

SDC/ SDE = Silent Data Corruption/ Silent Data Error
SLT = System-Level Test
VM = virtual machine
Q-pool = quarantine pool
Fleet = Data Center

1. License (Resilience Workstream)

1.1. Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in Open Web Foundation Modified Contributor License Agreement ("OWF CLA 1.0") ("Contribution License") by:

AMD, ARM, Google, Intel, Meta, Microsoft, NVIDIA

Usage of this Specification is governed by the terms and conditions set forth in **Open Web Foundation Modified Final Specification Agreement ("OWFa 1.0.2") ("Specification License").**

You can review the applicable OWFa1.0 Specification License(s) referenced above by the contributors to this Specification on the OCP website at http://www.opencompute.org/participate/legal-documents/. For actual executed copies of either agreement, please contact OCP directly.

Notes:

1) The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS,

IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.2. Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback:

TBD

2. Compliance with OCP Tenets

2.1. Openness

The SDC specification aims to remove barriers by providing standardized information exchange formats and an open framework in addressing SDC given the diversity of chips in the data center. There are three principles that enable openness, collaboration and contribution from the entire community.

1. Open Frameworks to advance the state of the art in SDC testing, enabling the community to make contributions on detection methods.
2. Standard format for both input and output to enable open sharing and collaboration.
3. Create communication pathways and transparency to accelerate our collective work

2.2. Efficiency

Contributes to the reduction of test infrastructure costs by providing a standardized information exchange format and open framework for faster and better SDC detection techniques. Failing to do it will resort to massive redundancy, which will hurt energy efficiency and performance of data centers. This Promotes energy efficient testing methodology by implementing optimized test content and efficient coverage of underlying hardware.

2.3. Impact

We are developing standardized formats and behaviors for sharing information about defective parts. This will enable researchers to develop better ways to protect data centers and open up new areas of research in hardware, software, and theory. In the past, experts in these fields have not been able to collaborate effectively due to limited data sharing. However, this is no longer sustainable. Standardizing data sharing will allow us to collaborate and create better solutions for data centers. This will enable us to codesign solutions across the system stack through a global supply chain, which will achieve the transformative impact that the Open Compute Project seeks.

2.4. Scale

SDC is an industry-wide issue impacting all platforms deployed threatening business workload and user data. The reliability of underlying hardware is critical to the future of artificial intelligence and machine learning. Furthermore, a single entity cannot solve the entirety of the problem considering how complex of a challenge it is and hence bringing the whole industry and academic community is key. Our contribution meets this scale tenet by defining standard information exchange formats and frameworks that enable adoption of new solutions and better detection methods through a global supply channel, accelerates reliable time-to-market advantage of technology, and supports ongoing innovation as well as legacy infrastructure.

2.5. Sustainability

Nothing can be sustainable if the fundamental tenet of computing is in jeopardy. So, any effort towards sustainability must overcome the SDC challenges. By ensuring there is a standardized format of information exchange and open framework to advance the state of art in SDC detection and mitigation across a wide variety of hardware, SDC specification can help improve the reliability and sustainability for future data center infrastructure.

3. Version Table

Date	Version #	Author	Description
June 30, 2023	0.3	See front page	Defines Scope and Inputs & Outputs for a test.

4. Scope

4.1. In Scope

As part of the Open Compute Project (OCP), in October, 2022 we established a Resilience working group (under the Hardware Management Project) focused on Silent Data Corruption (SDC) to create a community to address hardware-induced Silent Data Corruption challenges [Ref]. The scope of this initiative includes CPUs, GPUs, and other hardware accelerators. The founding members of this working group are AMD, ARM, Google, Intel, Meta, Microsoft, and NVIDIA.

The high level goals of the working group include:

1. Drive solutions and best practices that prevent and detect SDCs.
2. Create awareness about SDC challenges across the computing community.
3. Partner & engage with the academic community to actively address growing SDC challenges.

This specification document is intended to standardize test inputs, outputs, metrics, and formats to allow the industry and academia to collaborate on improving best practices in SDC testing.

The initial version of this document: defines an SDC-causing computing system; and describes the information needed in a test input specification, test output specification, and part history specification.

Future versions of the document will include (at least) complete input, output, and part history format specifications; and a set of metrics used to assess test effectiveness.

4.2. Out of Scope

This version of the document doesn't cover details on methodology and process for the research community to access faulty machines for experimentation. This also doesn't cover details on metrics and test framework, which will be covered in later versions of this document.

5. Definitions

5.1 SDC-causing Computing system definition

A faulty machine is categorized as SDC-causing if we can identify a (bug-free) workload (e.g., a test created by SDC screening tools or an actual application workload) that produces incorrect results without any indication from the built-in error detection mechanisms.

6. Introduction to Information Exchange Specifications

Sections 7-8 describe a standardized information exchange to enable collaboration across academic institutions, hyperscalers as well as silicon vendors and tooling communities.

It is important that the communication around the progress and results have the same definitions and a standard format across academia and industry. To enable sharing information using the same format amongst each other, we propose an information exchange format. This section is the first attempt at enabling standardization around the information shared related to an SDC causing computing system as defined in the *Definitions* section.

Information Exchange Format:

This is written with the intent for use in communication with academic institutions. We anticipate that this will lead to new techniques for overcoming the SDC challenge. Examples of such techniques include new system-level tests (that may be provided to fleet quarantine pools and Data Center (fleet) owners can provide standard outputs for the submitted tests) and new scan-based tests that may be applied to designs supporting fleet wide scan (or may be studied using simulations).

A standardized format allows for iterations across academic teams and also allows us to compare effectiveness of various techniques. There would be limitations across quarantine pool populations between new machines flowing into the pool as well as machines flowing out due to decommissioning of servers within a fleet.

Sections 7-9 are a list of key fields that describe what needs to be provided with a new test, what the output report will contain after testing is completed, and useful information about identified defective parts. The screens or detection methods the academic community develops will be run on a pool of defective machines. The goal is to establish a library of parts and its characteristics in a database to enable ease of access with information sharing.

7. Test Input Specification

Key fields:

1. **Tool configuration:**
    a. **Required**. Indicates which tool or tool suite to use and desired command-line arguments. Optionally, this may include specific version numbers or even specific builds (e.g., compiler choice, static vs dynamic linking, etc.). In the case of a suite of tools or a tool-running tool, all indirect tools need to be specified too.
    b. *Reason for this field*: to provide the entity executing the testing the necessary information on what to run to ensure proper reproduction.
    c. *Suggested practice*: specific binaries (if acceptable) or source code to be built in a container.
2. **Software environment selection**:
    a. Indicates what other software must be present in order for the tool to run (usually dynamically-linked libraries), operating system version and any necessary OS-level tweaks, such as whether to run with administrator/root privileges, changes to /sys or /proc/sys (Linux) or in the registry (Windows), etc.
    b. *Reason for this field*: to clarify the necessary permissions the tool will need as well as to limit variance caused by different ancillary software versions.
    c. *Suggested practice*: virtual machine image, container, or container-building instructions (e.g., Dockerfile), plus host configuration.
3. **Hardware environment selection**:
    a. Indicates the target desired hardware (see output fields below for recording) as well as any ancillary hardware that must be present, such as accelerators, amount of main RAM, and whether access to said hardware must be bare metal, virtualized, or if it is unimportant. If relevant, firmware versions to be installed.
    b. *Reason for this field*: for the target device, to specify what the desired test is; for others, to ensure they do not affect the results or to verify whether they do.
    c. *Suggested practice*: node network identifiers, serial numbers, Best Known Configuration settings.
4. **Desired run-time:**
    a. Indicates the run-time to ensure data collection accumulates sufficient samples.

The input may be as vague or as detailed as necessary. For example, it is not necessary to specify a specific operating system version and runtime if those aren't known or suspected to influence the results.

8. Test Output Specification

Key fields:

1. **Device type and device architecture:**
    a. Indicate at a high level what kind of computing device it is. E.g., processor, GPU, AI hardware, etc.
    b. *Reason for this field:* to provide a higher level device aspect to the research community. It can also showcase that the SDC problem isn't confined to specific kinds of devices.
2. **Technology Node:**
    a. Include technology node information
    b. *Reason for this field:* to provide a higher level overview on the technology node at which the problem manifests. This can aid to show that the SDC problem isn't confined to specific technology nodes.
3. **Tests applied during Vendor Manufacturing:**
    a. Provide high-level information regarding the type of tests (workload) which were run to determine the component as SDC-causing. This is limited to the tests executed as part of manufacturing testing.
    b. *Reason for this field:* Many researchers do not have exposure or access into numerous aspects of testing parameters during manufacturing testing. To broaden the understanding of manufacturing tests which did not catch these defective chips will be important for innovative solutions. Moreover, for time t > 0 degradation, the burn-in/stress test information could be important.
        i. *Scan tests:* coverage, test types, voltage, temp (at least qualitatively, hot/cold, high V, low V), frequency.
        ii. *Functional/system-level tests:* Similar fields as above.
        iii. *Burn-in and other stress tests*
4. **Tests applied post-manufacturing:**
    a. Provide high-level information regarding the type of tests (workload) which were run to determine the component as SDC-causing, these tests are limited to post manufacturing stages and in the datacenter. This would represent the tests or test suites that are run on machines to detect SDC-causing chips or machines in the datacenter as part of pre-deployment or *datacenter acceptance testing*. This field is to include all the **failing test names and test time per chip** to help with any correlation.
    b. *Reason for this info:* It will help a lot if the research community understands the type of tests which are failing and are effective in detecting SDCs. Providing high level data terms like system-level tests while failing to provide any insights would not yield research progress. Test times are important for the following reasons:

(a) for test content optimization from manufacturing through in-field testing, (b) understanding various trade-offs and system-level adaptation, (c) test length plays a critical role in estimating coverage metrics.

5. **Test environment:**
    a. Provides an overview of the test environment and the related system level conditions. For example: were these tests run on bare-metal or VM/container based environments.
    b. Associated kernel and firmware versions while the tests were executed if they could be shared publicly. Any associated device configs which could affect the environment and could be shared would be included within this field.
6. **Time to test failure:**
    a. Represents the time it took for the test to fail for a particular failure. It could also represent aggregate metrics, which quantify average test failure time per architecture per test type. This represents the time from which a test was initiated on a machine to the time towards first detection of an SDC-causing failure.
7. **Reproduction rate across test iterations (minimum 10 iterations):**
    a. Represents the ability for the test to detect a failure across multiple iterations, a minimum of 10 is noted here as an example. However, based on the failure mode and nuance associated with the failure a different iteration denominator could be used (and explicitly stated) to arrive at a better characterization. The threshold could be subject to further discussions or modifications based on feedback and testing efficacy.
8. **Error detection latency (if available):**
    a. It is the time elapsed between creation of an error due to a defect and its detection by various tests (workloads). It is understandable that tools that are available today within the industry cannot accurately quantify such a vital metric – the key to understanding the origins of the SDC problem and eventually creating solutions.

       This field has nuances associated as inherently there will be a delay between the true origin of defect and the time at which the defect is recognized. This field quantifies the risk of the delay in detection of SDC-causing defects within a chip.
    b. This has direct bearing on coverage, diagnosability (essential for test content coverage). It is also important for test times and detection efficiency.
    c. There are many ways of quantifying error detection latencies.
        i. Hardware support (e.g., signature analyzers)
        ii. Instrumentation of existing system-level tests with various granularities of checking.
        iii. Fine-grained checking in hardware or in software

iv. Fault-tolerance techniques

9. **High level failure info:**
    a. Represents a high level failure information regarding the test failure. This could be information related to functional block or the nature of the failure i.e hard crash on test vs fail-continue on test and other aspects associated with silicon. Include specific information on the tools that aided in catching the defect.
    b.
10. **Example log (if tool is open sourced):**
    a. Example log would be shared if the tool is open sourced or any specific tools log that a vendor is willing to share. This standardizes test development and enables automation of log parsing.

9. Part History Specification

Once a defective part is identified, the following key fields describe useful characteristics about the part that may help test authors understand the sources of failure and how to create better tests to find them. Because these details can reveal some vendor-proprietary information (e.g. manufacturing processes, failure analysis methods, failure rates), some vendors may choose to provide them only under a Non-Disclosure Agreement (NDA).

11. **Age at first failure:**
    Represents the age of the device at which the test failed for the first time. For testing within the quarantine pool, age may be a misleading factor. [Add a specific definition of what age means here]

12. **T = 0 failure or t > 0 degradation**
    Age at first failure would not provide a complete answer on the origin and manifestation of the failure. For example, a T=0 failure might be detected after 1 year (new tests with coverage, same test detected later due to specific microarchitectural states, etc.).

    Additional telemetry and understanding of coverage is required to truly characterize a failure as degradation at the chip level. [Add a comment on deterministic test] At the test level, this can represent if more defects are detected with time in fleet and repeated testing for the same content (or) if the failure manifested and was uncovered because of a novel test path.

    Reason: This information is important for understanding SDC mitigation strategies and testing costs within a large-scale fleet.

[Add a placeholder in the IEF - all the defined metrics are collected and added to error reports]

13. **Additional Diagnosis details as available from vendors** (assuming every chip passed manufacturing tests and failed system-level tests post-manufacturing)
    a. **13a** What kind of Failure Analysis was done and results
    b. **13b** Scan-based diagnosis
        i. What scan test detected it together with test conditions?
        ii. What were the fault diagnosis results?
            1. Timing-independent combinational defect
            2. Sequence-dependent (timing-independent) defect
            3. Timing-dependent
            4. E.g., Stuck-at, bridging, open faults (how many fault candidates)

10. Example of Test Output and Part History

Below is an example to show what information is intended to be shared for test output and part history. The goal is to build a database of this information on pools of machines referred to as quarantine pools (or Q-pools) that contain both known-bad and suspected-bad machines. More information about how Quarantine pools will be managed will be forthcoming in a future revision of this specification.

	Chip #		1	2
1	**Device type/architecture**		CPU	CPU
2	**Tech node**		7nm	14nm
3	**Tests applied during manufacturing (Vendor to provide)**	**Scan tests (type, coverage, test conditions) that caught the failure**	N/A	N/A
		Functional tests (type, coverage, test conditions) that caught failure	Functional	Assembly + C (functional)
		Burn-in & other stress tests	N/A	N/A
4	**Tests applied post manufacturing together with test conditions (SLT)** **4a. Test name** **4b. Corresponding durations** **4c. Test condition**		4a. CPUcheck 4b. 10 min 4c. NA 4a. Rainbow 4b. 10 min 4c. NA	4a. X87 test 4b. 1 min 4c. NA 4a. Test <name> 4b. 5 min 4c. NA
5	**Test Environment:** **Tests run on bare-metal or VM/container based environments**		Bare-Metal	Bare-Metal
6	**Time to Test Failure** **6a. Tool that triggered failures** **6b. corresponding time**		6a. Rainbow, 6b. 36 min	6a. X87 Test, 6b. 10s-1 min
7	**Reproduction rate of system-level tests and environment**		100% @ Q-Pool	100% @ Q-Pool
8	**Error detection latency**		NA	NA
9	**High-level failure info**		Rainbow (Tool specific, Seed# etc)	X87 test, instruction specific failure
10	**Example log**		Attach the file	Attach the file

11	**Age at first failure** **11a. SLT tests during the machine life** **11b. For Scan based test, include if its the same**		11a. T=0 at QA (eg1), SLT - rainbow 11b. T = 18 mon, scan test xx	11a. T=11 mon, first scan of the test
12	**T = 0 failure or T>0 degradation** **(when fleet scan testing is enabled, this information should be available)**		T= 0 failure, T>0 NA	T= 0 failure, (failed at first test iteration) - higher prob. of T=0 failure. T > 0 NA
13	**Additional Diagnosis details (as available from vendors)**	**Failure Analysis:** **(a) Failure Analysis performed & Results?** **(b) Reproduction using scan testing** **(c) Test detected with conditions, if any** **(d) Fault diagnosis - e.g. timing dependent, timing independent, stuck-at, sequence dependant etc.**	-	Reproduced at vendor, Test added to MFG, Instruction failure at FPU stack to normal stack transfer (marginal timing scenario)

SLT = System-Level Test
VM = virtual machine
Q-pool = quarantine pool

11. References (recommended)

[1] TBD

Appendix A - Checklist for IC approval of this Specification

Complete all the checklist items in the table with links to the section where it is described in this spec or an external document .

Item	Status or Details	Link to detailed explanation
Is this contribution entered into the OCP Contribution Portal?	Yes	If no, please state reason.
Was it approved in the OCP Contribution Portal?	Yes or No	If no, please state reason.
Is there a Supplier(s) that is building a product based on this Spec? (Supplier must be an OCP Solution Provider)	No	This is a 0.3 draft for feedback from the Community. No products will be built from this version.
Will Supplier(s) have the product available for GENERAL AVAILABILITY within 120 days?	No	