ReLiANT

Team Standard

November 9, 2025

For

SanDisk

Faculty Mentor: Ogonna Eli
Team Members: Xavier Graham, Fermin Valenzuela, Brody England, Isaac Schwarz

Version 0.1

Accepted as baseline requirements for the project:
For the client:___Chris Ortiz 11/7/25___________________________
For the Team:_______Xavier Graham 11/7/25_______________________

Table Of Contents

Introduction

The data-storage industry is one of the most competitive and innovation-driven segments of modern computing. Solid-state drives (SSDs) have become the dominant medium for persistent storage across consumer electronics, enterprise servers, and hyperscale cloud infrastructures, with the global SSD market exceeding 60 billion USD in annual revenue and continuing to grow as demand for high-speed, low-latency storage accelerates. Central to this ecosystem is the NVMe (Non-Volatile Memory Express) standard, which defines a unified protocol for accessing non-volatile memory over PCI (Peripheral Component Interconnect) Express interfaces. NVMe delivers parallelism and queue-based access that allow storage systems to achieve unprecedented throughput, but this complexity also introduces significant challenges in ensuring correctness, performance, and compliance across firmware, drivers, and hardware implementations.

Within this industry, SanDisk Corporation develops storage technologies that power consumer devices and enterprise products worldwide. SanDisk's *Tools and Infrastructure* division, led by Senior Technologist Chris Ortiz and Vice President Rex Jackson, focuses on the internal frameworks and testing platforms that verify drive behavior against NVMe specifications. Ensuring that each new product generation fully conforms to the standard is critical for interoperability and reliability. Traditional validation workflows depend heavily on manually written test scripts produced by Test Engineers. While effective, this process can be labor-intensive, prone to human oversight, and limited in the diversity of test sequences it explores.

The ConTiNGENT project-short for *Convert TLA+ into NVMe Generative Test*-addresses these limitations by integrating *formal specification* and *automated test generation* directly into SanDisk's validation pipeline. Building upon earlier proof-of-concept work under Western Digital's *Generative Testing of SSD using TLA+ Model Simulation* initiative, this new effort re-implements the system with a more robust architecture. Instead of relying on PlusPy, the prior Python-based interpreter, ConTiNGENT utilizes TLC, the official TLA+ model checker, to explore all nondeterministic behaviors defined in the specification and generate executable traces. These traces are then bound to the real NVMe command interface using Rust and the open-source *nvme-cli* toolkit.

This collaboration with Northern Arizona University exposes students to state-of-the-art formal methods and low-level device protocols. By authoring and instrumenting TLA+ models that mirror the NVMe Base Specification, the team will create a bridge between high-level correctness reasoning and concrete drive behavior. The outcome will be a reusable framework capable of generating deterministic, specification-compliant test sequences that improve coverage, reproducibility, and efficiency compared with manual or randomized testing. Ultimately, ConTiNGENT represents a forward step toward specification-centric verification-an approach where the specification itself becomes the engine of validation-helping SanDisk 2.0 evolve its testing process for the next generation of NVMe products.

Problem Statement

As mentioned above, NVMe drives need to go through rigorous testing and quality verification before they can be released to consumers. Currently, at SanDisk test engineers manually design test sequences based on NVMe specifications. They use TLA+ to write specifications to list how a NVMe drive should operate, then they use the nvme-cli tool to run commands to verify whether the drive behaves as expected. Engineers analyze the results and refine test cases iteratively to improve test coverage and reliability. While this process creates a controlled testing environment, it is resource heavy, slow to adapt and dependent on manual testing. This testing approach is inefficient and presents the following drawbacks:

- **Limited Test Coverage**:
  - Manually authored tests cannot feasibly capture all possible state transitions and command interactions defined in the NVMe specification. As a result, certain edge cases remain untested.

- **Decreased Bug Detection:**
  - Because engineers tend to design tests based on anticipated behaviors, defects that occur in less obvious or emergent conditions are often missed.

- **Test Noise:**
  - Test noise is the term used to describe when a test issues a false negative, this causes an engineer to investigate the false positive. Manual testing can create a lot of test noise.

- **Lack of Automation and Specification Integration**:
  - The current testing process is decoupled from the formal NVMe specification. There is no direct mechanism to translate formal TLA+ system models into executable test sequences.

SanDisk's existing SSD testing workflow is constrained by manual testing that fails to scale with the complexity of NVMe drives. The lack of a formal bridge between TLA+ specifications and executable test sequences leads to inefficiencies and wasted resources. To address these challenges, we plan on building a generative test framework that automatically converts TLA+ models into NVMe commands, enabling broader coverage, improved bug detection, repeatability, and a more efficient testing pipeline.

Solution Vision

Our proposed solution, the ContiNGENT (Convert TLA+ into NVMe Generative Test) project, is an automated framework that bridges the gap between TLA+ specifications and NVMe device testing. This system will automatically generate and execute NVMe test sequences that are derived from formal TLA+ models, this removes the need for manually authored test cases. By using TLC which is the native TLA+ model checker with a Rust based engine, ContiNGENT will ensure that every valid system behavior defined in the NVMe specification is represented, executed, and verified on real hardware. This method will transform how SanDisk currently conducts their testing and will result in a less time-consuming process and transform it into an automated, specification-driven workflow that enables broader test coverage, reduces false negatives, improves bug detection, and increases testing efficiency.

Key Features

- **Automatic Test Sequence Generation**: Uses TLC's generate mode to produce valid NVMe command sequences directly from TLA+ models.
- **Rust-Based Command Execution**: Converts generated sequences into executable commands using the nvme-cli interface, to provide safe and efficient hardware testing.
- **Repeatable, Deterministic Testing**: Generates replayable sequences to reproduce and analyze specific test outcomes.

System Data and Workflow

- **Input Data:**
  1. TLA+ specifications defining NVMe command behaviors, states, and transitions.
  2. Configuration details from the NVMe DUT (Device Under Test).

- **Generated Data:**
  1. Valid NVMe test sequences.
  2. Execution logs and validation reports comparing expected vs. actual behavior.

- **Core Processes:**
  1. **Model Exploration:** TLC explores the TLA+ model to generate valid command sequences.
  2. **Sequence Translation:** Generated sequences are formatted into executable commands.
  3. **Test Execution:** Rust-based executor issues commands to the SSD using nvme-cli.
  4. **Verification:** Device responses are validated against TLA+ state expectations.
  5. **Reporting:** Results are stored, analyzed, and used for retesting if necessary.

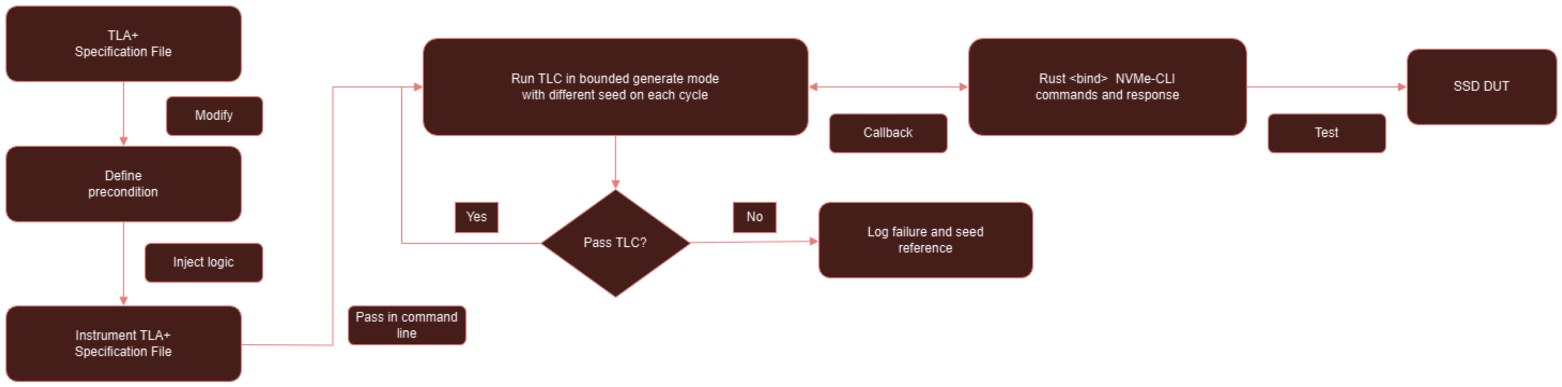
TLA+ Specification File
Modify
Define precondition
Inject logic
Instrument TLA+ Specification File
Pass in command line
Run TLC in bounded generate mode with different seed on each cycle
Yes
Pass TLC?
No
Log failure and seed reference
Callback
Rust <bind> NVMe-CLI commands and response
Test
SSD DUT

Figure 1. Architecture Diagram of proposed solution

Broader Impact

ConTiNGENT moves SanDisk toward formal-model-driven testing, where specifications themselves become the source of truth for verification. This reduces human error, accelerates validation, and provides a framework that can be expanded to other hardware or software systems beyond NVMe. Ultimately, it establishes a scalable, intelligent testing model that could redefine how device verification is performed across the industry.

Project Requirements

This section outlines the specific, measurable requirements for the ConTiNGENT (ReLiANT) project. These requirements are derived from the Solution Vision and are technically grounded by the Feasibility Analysis. All functional, non-functional, and validation requirements will be prioritized using the MoSCoW method.

1. Functional Requirements (FR) - describes what the system must do.

1.1 Must-Have Functional Requirements

FR1 — Input Handling (**Must**)
The system shall accept a TLA+ specification output file as its primary input.

FR2 — TLC Simulation (**Must**)
The system shall use the TLC model checker to simulate the TLA+ specification and generate deterministic execution steps.

FR3 — Trace Parsing (**Must**)
The system shall parse the TLC output to extract a sequence of actions and states.

FR4 — Action Mapping (**Must**)
The system shall map abstract TLA+ actions to concrete NVMe CLI commands using a user-defined YAML or JSON configuration file.

FR5 — DUT Execution (**Must**)
The system shall execute the generated NVMe-CLI command sequence on a Device Under Test (DUT).

FR6 — Logging & Recordkeeping (**Must**) system shall log the following for each test sequence:

- the original TLA+ trace
- the generated NVMe commands
- stdout/stderr from nvme-cli
- the final pass/fail result

FR8 — Safety Enforcement (**Must**)
The system shall implement safety checks preventing execution of high-risk NVMe commands (e.g., FormatNVM, Sanitize) unless explicitly overridden.

1.2 Should-Have Functional Requirements

FR7 — Dry-Run Mode (**Should**)
The system shall provide a dry-run mode that prints the generated command sequence without executing it.
FR9 — Long-Run Automation (**Should**)
The system should support unsupervised execution of test sequences for extended periods.

2. Non-Functional Requirements (NFR) - defines how the system should behave and the qualities it must maintain.
2.1 Must-Have Non-Functional Requirements

NFR2 — No Indefinite Hangs (**Must**)
All test sequences must have a configurable timeout. If the timeout is exceeded, the sequence is marked as failed and the system continues.

NFR3 — Safety by Default (**Must**)
The system's default configuration must prevent commands that risk data loss or device damage unless explicitly opted in.

NFR6 — Test Coverage for Correctness (**Must**)
The Rust execution layer must maintain high unit and integration test coverage to avoid false positives or negatives caused by internal logic errors.

2.2 Should-Have Non-Functional Requirements

NFR1 — Performance (**Should**)
The system should generate and begin executing a test sequence within 60 seconds for moderately complex TLA+ models.

NFR4 — Minimal Setup (**Should**)
The system should run from the command line with minimal setup (Rust, Java/TLC, nvme-cli).

NFR5 — Maintainability (**Should**)
The source code should be modular and documented to support extension of action mappings and command behaviors.

3. Validation & Verification Requirements (VR)

These requirements define how system correctness and reliability will be evaluated.

3.1 Must-Have Validation Requirements

VR2 — Model Accuracy Verification (**Must**)
The system must generate test sequences that accurately reflect the TLA+ model, validated against a known-good simulated DUT.

3.2 Should-Have Validation Requirements

VR1 — Full Traceability (**Should**)
The system must allow an engineer to trace any NVMe command failure back to the originating TLA+ trace step
.
VR4 — Graceful Error Handling (**Should**)
The system must log NVMe command failures and mark the test sequence as failed without crashing.

3.3 Could-Have Validation Requirements

VR3 — Coverage Reporting (**Could**)
The system could generate a coverage report showing which TLA+ states and transitions were exercised during testing.

Potential Risks

ConTiNGENT's purpose is testing NVMe drives for Sandisk, so risks include false test results, wasting time, and even destroying drives permanently. This section will break down these potential risks, and what ReLiANT is doing to prevent them.

False test results can occur when doing generative tests due to logical errors in the program logic. Incorrect mapping of state between the test spec and the physical drive could cause a false negative or false positive to be reported. False negatives waste time, as engineers will have to spend time looking into the results to verify whether the testing suite failed, or the drive is actually faulty. False positives could result in faulty drives being sold to customers, which would reflect poorly on Sandisk.

Here are some ways failures could occur and how ReLiANT will be handling them.

- Logical errors in the Rust source code.
    - Uncaught edge cases and incorrect program logic in the Rust code could cause the physical drive to be issued incorrect commands, which could cause it to return unexpected results. Unexpected results are bad as our test sequences are discrete state machines that only have one valid path of execution. If the drive deviates from what is expected, it will report a false negative. Additionally, although rarer, incorrect commands issued from the Rust execution layer could cause a test that should have failed to pass. This would report a false positive. In short, faulty Rust logic could cause false test results, positive and negative, and need to be prevented.
    - A way to mitigate this problem would be unit and integration tests for the Rust code. A solid test suite for the Rust layers will increase the clients confidence in the accuracy of the test results reported.
- Incorrect specification.
    - If the TLA+ specification describing the NVMe device does not accurately describe the behavior of the real device, then incorrect test steps may be generated.
    - The way to mitigate this problem is to rigorously study the NVMe specifications and consult trusted sources to ensure specifications accurately describe the behavior of the DUT's.

- Test suite hanging.
    - ConTiNGENT is meant to run unsupervised for long periods of time, testing DUT's without the intervention of Sandisk engineers. If the test suite runs into an

    infinite loop during execution, it could run uselessly and unnoticed for potentially long periods of time. This wastes time and computing power.
    - Timeouts to end very long running test sequences will mitigate the wasted time.

- Dangerous NVMe-CLI commands.
    - NVMe-CLI is a linux program used to issue low level commands to NVMe drives. Some of these commands are potentially dangerous, and can temporarily or permanently break drives. Admin commands like FormatNVM, Sanitize, and admin-passthru can cause loss of data on the drive that can brick the device.
    - To avoid this, ReLiANT will determine risky scenarios that are relevant to the tests we are running, and implement safety features that prevent these scenarios from occurring.

Project Plan

The important milestones in the development of this project as it stands now are:

Milestone Description	Completion Date
Completion of a usable TLA+ specification.	November 2025
Prototype Rust program capable of parsing TLC output and executing the appropriate commands on a real NVMe drive.	November 2025
Basic logging. Must report the result of the test, the test sequence, and highlight failing states.	November 2025
Unit/Integration test suite.	February 2026
Refinement of specifications, Rust runner, and logging. To the client's specification.	December 2025 - May 2026
Automatic Retest (stretch goal). Implement automatic retest logic for failing tests.	March - May 2026

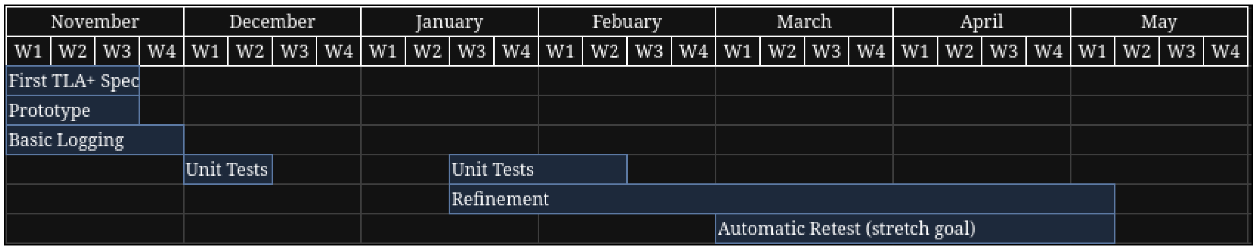

Figure 2. GANTT chart of the milestones.

Conclusion

The ConTiNGENT project presents a robust and technically-sound solution to a critical challenge in SanDisk's NVMe drive validation process. By automating the bridge between formal TLA+ specifications and physical hardware testing, the project directly addresses the inefficiencies, limited coverage, and potential for human error inherent in manual test creation.

The requirements outlined in this document define a clear path forward. The selected architecture—leveraging TLC for reliable model simulation, Rust for a safe and efficient execution layer, and a declarative JSON configuration for maintainability—is proven feasible through initial demonstrations. This approach ensures the system will be capable of generating comprehensive, deterministic test sequences that significantly enhance test coverage and bug detection while minimizing disruptive test noise.

Furthermore, the project incorporates key risk mitigation strategies from the outset. Safety checks against destructive commands, configurable timeouts to prevent hangs, and a focus on code correctness through rigorous testing will ensure the framework is both safe for valuable hardware and trustworthy in its results.

In summary, ConTiNGENT is designed to transform SanDisk's testing workflow from a manual, resource-intensive process into an automated, specification-driven powerhouse. Successful implementation will not only improve product reliability and reduce time-to-market but also establish a scalable, formal-methods-based testing paradigm that can serve as a benchmark for future validation efforts. This project delivers not just a tool, but a foundational advancement in how storage devices are verified.