


# CS486C – Senior Capstone Design in Computer Science

## Project Description

Project Title: ConTiNGENT (Convert TLA+ into NVMe Generative Test)	
	<p><b>Chris Ortiz</b>, Senior Technologist Tools &amp; Infrastructure SanDisk Corp., Engineering &amp; Product Management chris.ortiz@sandisk.com</p> <p><b>Rex Jackson</b>, Vice President Tools &amp; Infrastructure SanDisk Corp., Engineering &amp; Product Management rex.jackson@sandisk.com</p>

### Project Overview:

In the original project under old Western Digital, we previously collaborated with the SSDynamics team as part of the NAU CS Capstone project. The motivation was to generate test sequences from a TLA+ specification—sequences not derived from any manually written tests by Test Engineers. The project was titled *Generative Testing of SSD using TLA+ Model Simulation*. This approach offered several advantages:

- **Bug discovery:** It could uncover bugs through test sequences that Test Engineers might not anticipate.
- **Broader test coverage:** Each iteration could generate different sequences due to the non-determinism in the TLA+ specification.
- **Resource efficiency:** It justified reduced DUT (Device Under Test) allocation and enabled more meaningful use of idle DUTs and test systems, compared to repeatedly running the same test sequence.
- **Noise reduction:** It avoided test noise common in randomized testing, which may execute sequences not allowed by the specification.
- **Repeatability:** Unlike fuzzing, the generated sequences were replayable, aiding in issue reproduction.
- **Specification-centric testing:** In the extreme case, Test Engineers wouldn't need to write test sequences manually but could instead focus on composing TLA+ specifications that help them deeply understand the industry specification.

SSDynamics successfully delivered proof-of-concept, though some results were unreliable. This was later traced to the use of PlusPy—a Python-based TLA+ interpreter—which we had proposed for model simulation.

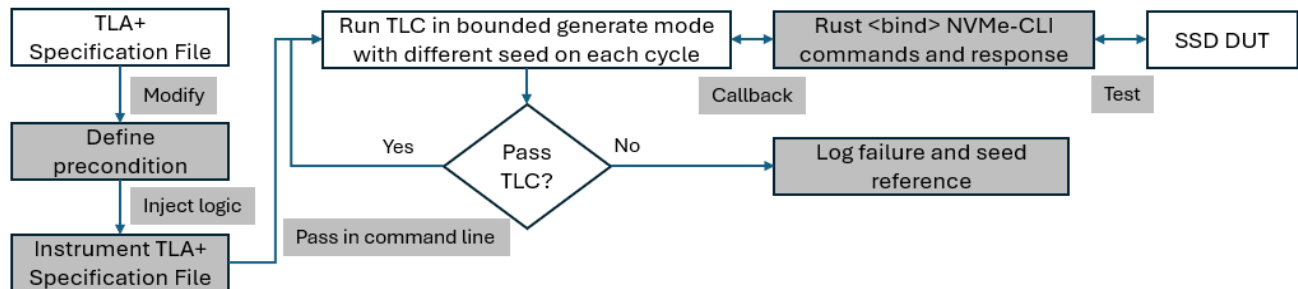
Under the new SanDisk2.0, we'd like to evolve this concept into **ConTiNGENT**, integrating it as part of our standard test process. The motivation remains the same, but we aim for a more robust and scalable approach.

We are requesting NAU's support to reimplement this project from scratch, with the following improvements:

- **Use of TLC:** Instead of PlusPy, we will use TLC—the native model checker for TLA+—which now supports a "generate" mode capable of handling all non-deterministic expressions.
- **Rust-based execution:** We plan to use Rust (instead of Python) to bind with NVMe-CLI for executing tests.
- **Specification expertise:** NAU will need to understand and write TLA+ specifications based on the latest NVMe Base Specification.
- **Instrumentation convention:** We expect NAU to define a convention for instrumenting TLA+ specifications to map conceptual states to the physical states of the NVMe DUT and system.

The figure below is our proposed solution where the grayed boxes and arrow's label are the ones we need NAU to help. This project will expose the NAU students to be familiar with LaTeX (Lamport TeX), PlusCal and TLA+ which Leslie Lamport invented. He is an ACM Turing Awardee where his contribution to Computer Science is used in universities, and by big companies like AWS, Microsoft, Oracle, Google, LinkedIn, MongoDB, Datadog, Intel, ARM, Nvidia, etc. Also, there is a potential that projects such as this can get grant from TLA+ Foundation if submitted and the committee selected it. The students will also be exposed to NVMe commands that can be executed to SSD (Solid-State Drives).

As a stretch goal, NAU can add logic for automatic retest of failing test sequence.



### Knowledge, skills, and expertise required for this project:

- Knowledge of TLA+ and PlusCal using TLA+ VSCode extension.
  - Home Page: <https://lamport.azurewebsites.net/tla/tla.html>
  - Github: <https://github.com/tlaplus>
  - Conferences: <https://conf.tlapl.us/home/>
  - Foundation: <https://foundation.tlapl.us/>
- Knowledge in issuing NVMe command to SSD:
  - Github here: <https://github.com/linux-nvme/nvme-cli>;
  - Ubuntu manpages here: <https://manpages.ubuntu.com/manpages/mantic/man1/nvme.1.html>
  - NVMe Base Specification: <https://nvmexpress.org/specification/nvm-express-base-specification/>
- Coding skill in Rust (preferred) or Python: <https://www.rust-lang.org/>

### Equipment Requirements:

Computer system with VSCode installed with the following extensions and internet connectivity.

- Rust-analyzer (The Rust Programming Language)
- TLA+ (Temporal Logic of Actions by TLA+ Foundation)
- Graphviz Interactive Preview (tintinweb)
- Live Share (Microsoft)
- PDF Viewer (Mathematic Inc)
- Desktop PC with secondary target SSD that is NVMe and PCIe Compliant.
- Latest Ubuntu

Other software:

- OpenJDK >= 11.0.6 (Please do not use Oracle's)
- Adobe Acrobat Reader
- Rust Installation and crates that might be needed

### Software and other Deliverables:

Detailed, clear, and professionally composed documentation and literature which SanDisk product teams will use as reference manual.

Complete and well-commented codebase via ZIP file format.

Github private account for sharing development codes.