

# Solid State Disk Simulator — Development

## Client

<Undisclosed company>, a global leader in consumer electronics, home appliances and mobile communications. With over 50 years of history, the company controls over hundred local subsidiaries, several dozens factories and R&D laboratories with about 80,000 employees worldwide.

## Business Need

The Customer was considering development of a Solid State Disk (SSD) solution for its laptop computers. A solid-state disk (SSD) is a data storage device that uses solid-state memory to store persistent data. An SSD emulates a hard disk drive interface, thus easily replacing it in most applications. Some of the attractive features of SSDs include silent operation due to the lack of moving parts, low power consumption and lower risk of mechanical failure. Many high-end laptops of leading companies are shipped with SSDs rather than HDDs, as this technology is expected to gain popularity in the nearest years.

Prior to this large-scale and time consuming project, a pilot sub-project - **SDD Simulator** - was initiated as its auxiliary part that would serve as the evaluation platform for the Flash Translation Layer (FTL) algorithm that is the core of SSD solutions. The Client planned to use SDD Simulator for performance evaluation and analysis of different system configuration under the realistic environment.

Auriga was handed this task, being a recognized software R&D services provider with proven expertise in embedded and mobile software development, unmatched performance level and well-known client oriented approach.

## Objectives

The goal of the SDD Simulator project was to develop the **Solid State Disk (SSD) simulation platform**. The purpose of the platform was to model the environment that an FTL algorithm is surrounded by, namely, the Flash (NAND) devices (that is used by FTL in order to store the data) and the Host (that supplies the FTL with a set of host commands). The SSD simulator should support several FTL algorithms.

Later, the FTL algorithm was to be implemented on a custom ARM-based board, and it was required that the

chosen FTL implementation should allow further porting to the target environment.

## Solution

Auriga developed the SSD simulation platform and implemented a reference FTL algorithm that allows iterative platform testing. The platform consists of the following components and features:

- **GUI Application** responsible for the SSD simulator components configuration, monitoring and control, along with data analysis and visualization. It analyses various SSD simulator data such as host command's frequency & distribution and represents them in a graphical form.
- **FTL Model**, capable of working with multi-bus and multi-chip Flash model, simulating the Flash Translation Layer that translates the Logical Block Addresses (LBA) into the Flash Physical Block Addresses (PBA). Its main features are the following:
- **Address Mapping (AM)** that maps the Host Logical Address to the Flash Memory Physical address. The mapping should be constantly changed due to the Flash memory nature (overwriting is prohibited).
- **Wear-leveling (WL)** to make the Flash write evenly to extend its lifetime. The Flash devices has rather small limit of the erase/write cycles. To keep track of the erase count of every block the wear-leveling algorithm will use the spare OOB area of NAND flash device.
- **Defect Management (DM)** to handle flash defect blocks. will handle both factory defects and growing defects detected during the life cycle of the Flash device in case of read, write, or erase errors. Write and erase errors will be reported by the NAND device.
- **NAND model** simulating the array of the Flash devices used by the FTL algorithm as the underlying storage. NAND model supports:
  - SLC, MLC and ONFI
  - multi-bus and multi-chip configuration
  - behavior replication of the real flash devices with respect to program, erase and other operations.
  - tuning of various parameters such as chip size, block size and page size, Flash memory access time, etc.

- tracking the read/write count of individual page
- **Host Model** was implemented to simulate the realistic behavior (in terms of I/O requests) of a Windows system with running applications inside. Host Model can operate in several modes to capture and analyze the host commands issued to a disk and/or generate a set of commands or reproduce the already captured set on the SSD model. Host Model is comprised of the following parts:
  - **Capture Program** captures and writes down the read/write commands to/from a hard disk when an application is executed.
  - **Test Vector Generator** generates an input for the FTL model from previously captured Test Vectors or generates them following patterns. Test Vector Generator is a user-space program that can be executed to playback the host commands previously captured by the Capture program and to generate the I/O requests.
  - **Virtual Disk Driver** (based on imdisk) represents the SSD model as a Virtual Disk for the Windows system. It provides an interface of a regular Windows disk driver and an interface to be used by the Test Vector Generator to execute generated I/O requests.
  - **User-Space Test Framework** was created to allow debugging this code in the user-space. This User-Space Test Framework will be a user-space library that implements the same interface that is used by the FTL and NAND code in the kernel environment.
  - **InstallShield-based program** was developed to install and configure the SSD Simulator components

Auriga successfully implemented simulation engine of SSD platform. Acceding to feedback from customer, the SSD Simulator was used for performance evaluation and analysis of different system configurations.

## Tools and Technologies

- Microsoft Foundation Classes (MFC)
- C/C++
- Windows XP Driver Development Kit (DDK), Microsoft Visual Studio
- imdisk, nandsim