Open Source SLC NAND Flash

Failure Analysis Platform

By:

Garrett Thomas, Greg Bray, Jeffrey Gorton, Jonathan Morgan, and Kyle Stewart

Date: April 4, 2008

Clinic Sponsor:
Micron Technology, Inc.

Advisor:
Ken Stevens
University of Utah

TABLE OF CONTENTS

ABSTRACT

The use of NAND Flash has increased worldwide each year as the cost of manufacturing decreases and memory density increases. Unfortunately, as manufacturing processes improve, the reliability and failure rates of NAND Flash memory have remained unchanged. Current SLC NAND Flash memory is guaranteed to operate for up to 100,000 Program-and-Erase cycles, but failure rates beyond this point are unknown since they will vary significantly between manufactures and product lines. The lack of information beyond 100,000 cycles has been a major barrier to adoption in a wider market of devices. A testing platform that can run a specified use-pattern, detect and record memory failures, and then return a graphical report of actual failure rates would be of great benefit to product engineers interested in using NAND Flash memory in their devices. By developing an open platform for NAND Flash failure analysis, we aim to improve the use of NAND Flash technology in future devices and provide reliable data on failure rates beyond the SLC cycle limit specifications.

INTRODUCTION

NAND flash memory has become the preferred choice for high-density, nonvolatile memory storage in applications such as digital photography, USB flash drives, embedded systems, mp3 players, cellular phones, and solid state drives. Because of its small size, low cost, and low power requirements NAND flash is frequently used to replace or complement magnetic disk, static ROM, or optical drive storage. New applications for NAND flash memory continue to emerge each year, and the total market for NAND flash memory is predicted to reach $15 billion by 2010 [1].

Due to the specific nature of flash technology, permanent and temporary failures may occur during regular use and will increase in frequency over the lifespan of the memory, eventually leading to complete memory failure. Failure can occur on a single bit or over an entire block and can be exacerbated by specific use patterns (partial page programming, high block read cycle count, high block erase/program count) [2]. Manufactures currently guarantee error-free functionality of their SLC NAND flash devices for up to 100,000 program and erase cycles when using basic Error Code Correction (ECC) algorithms [3]; however very little data is available regarding failure rates and usability past the manufacture specified limits as each unique NAND design, vendor, and fabrication process will demonstrate different failure mechanisms and behaviors.

The goal of this project was to create a simple and low cost platform for empirically determining the failure behavior of a SLC NAND flash chip beyond the manufacture specified limits. The system will allow users to define a custom use-pattern for wear testing over a specific number of cycles and will report the bit and block failure behavior measured under these conditions. This will allow product designers to make better use of NAND flash technology by

understanding the cycle limitations and planning for graceful device degradation using bad block retirement and ECC algorithms to obtain the desired product lifespan.

Also the entire testing platform is to be released as an open source project to help improve the current state of NAND flash testing for the entire industry. A Google Code website has been setup to store all of the project files and act as a permanent public presence for an ongoing community of users [4]. This will allow future users to expand the project to work with additional NAND components not covered by the initial release. This paper will discuss how this testing system was designed and implemented and outline each step of the process. It will also discuss the procedures used for testing the system to ensure accurate function and operation.

PROJECT TASKS

We divided the project tasks according to the various components and assigned team members to each task:

- Greg – Front-end GUI and Database
- Kyle – Firmware and NAND Flash Controller
- Jon – System Architecture (Verilog) and Database
- Garrett – Random Number Generator and Debugging/Quality Assurance
- Jeff – Documentation, NAND Flash Controller and PCB Board/Daughter Card

Each team member was responsible for developing and documenting the interfaces used by the components they were assigned to. The System Architecture involves the overall view of how the various Verilog components connect and interface to each other on the FPGA. It also includes the pin mappings that connect the Verilog flash controller through the daughter card and to the flash device itself.

SPECIFIC TASK INTERFACES

This section describes how the various components interface with one another. The nine major components with specific interface requirements are listed in the table below and range from the user interaction with a front-end graphical user interface (GUI) to the hardware controlling the flash module itself. The interfacing model follows the flow of information from the high-level user input to the low-level implementation hardware. It describes the interfaces in a top-down manner beginning with the high-level GUI. Commands flow through the interfaces until they are finally executed on the NAND flash hardware. The results of these executions then flow back up the chain of interfaces and are stored in a database. The user can then view reports using the application GUI.

Component	Physical location	Technology/Platform
SQL Database	Host Windows PC	Microsoft SQL Server
Front-end GUI	Host Windows PC	Microsoft .NET C#
USB 2.0 Driver	Host Windows PC	libusb-win32 and LibUsbDotNet
USB 2.0 Driver	Altera DE2 FPGA	Phillips ISP1362 USB controller
C Firmware	Altera DE2 FPGA	Altera Nios II IDE
NAND Flash Controller	Altera DE2 FPGA	Verilog, Altera Quartus II
Daughter Card	Daughter Card (DC)	PCB provided by Micron
NAND Flash Module	Daughter Card (DC)	SLC provided by Micron

Table 1: List of project interfaces

User <=> Front-end Graphical User Interface (GUI) on PC

The GUI allows the user to customize various parameters that affect how the program tests the NAND flash. The user chooses various testing techniques and verifies the results. The GUI allows the user to execute the following set of basic commands.

- Erase a block
- Check block for erase failure flag
- Report the number of erasure failures (i.e. the number of 1's in an erased block)
- Program a page
- Verify programming was successful
- Read Page
- Read Verify - compare/verify a block against originally programmed pattern

The GUI also supports the ability to allow any sequence of the above commands to be automated. This can be done through the GUI itself or through a scripting language. In addition to the basic command set, the GUI can also allow the user to specify the following parameters.

- Range of blocks to be tested
- Number of cycles to repeat the command (minimum range of [1, 1048575])
- Auto-generated random number generation or a specific pattern

Front-end Graphical User Interface (GUI) on PC <=> USB 2.0 driver on PC

In order to interface the host to the USB 2.0 host driver, the libusb-win32 library version

0.1.12.1 was used. This libusb-win32 library is a Windows port of the libusb generic usb driver for Linux/UNIX systems. This library serves as a wrapper to a generic USB driver for Windows XP. Using this library, we were be able to talk to the bus using code developed and executed in user space and avoid the potential hazards of developing a custom, kernel space driver. There is an actively developed C# wrapper for the libusb-win32 library called LibUsbDotNet [5]. This wrapper allowed us to develop our GUI code on the C# .NET platform and interact with the device through USB without using unmanaged code or native device drivers.

Frontend GUI <=> SQL Database

The GUI then collected the results of each test by using the USB interface described above. These results were then stored in a SQL database for further processing. The GUI communicated with the database using the standard ActiveX Data Objects classes of the Microsoft .NET framework (ADO.NET). The GUI then used Transactional SQL (T-SQL) to query the information stored in this database and present to the user the testing results. Initially the SQL table contained a single table with six columns holding the following information:

- ID - A unique identification number assigned every time a new instruction is performed
- Cycle - The cycle count for this instruction
- Memory Address - The place in memory where this instruction begins
- Function Name - The name of the instruction performed (read block, erase block, program, etc.)
- Status - Pass/Fail

Eventually, the table will be expanded to include more fine grained detail that provides more

insight into the failure characteristics of flash device.

C Firmware on the FPGA <=> USB 2.0 driver on the FPGA

The firmware running on the FPGA was written in the C language and compiled to run on the Nios II soft processor available for the DE2 FPGA development board [6]. The firmware interfaced with the Phillips ISP1362 USB controller chip built on the DE2 board. This board is compatible with the USB 2.0 specification, but only communicates up to full-speed (12 Mbits/sec). The firmware was able to process packets according to the USB 2.0 specification as well as interface with the libusb-win32 and C# code written for the GUI. The firmware received a 32 byte command from the GUI and send the results back to the host (up to 2112 bytes for read page) using as much of the 12 Mbits/sec USB bandwidth available to it.

C Firmware on FPGA <=> NAND Flash Controller on FPGA

The FPGA firmware interprets the commands sent to it through the USB 2.0 interface and issues them to the NAND Flash Controller. This involves listening for communication from the host and properly parsing the 32 byte commands into a sequence of instructions for the NAND Flash Controller. After the command has been parsed, the firmware then communicates those instructions to the NAND Flash Controller through the on-chip RAM buffer. The processor then waits for the command to execute on the flash and gather any results it needs. This information is then sent back over the bus through the USB interface to the host.

NAND Flash Controller on FPGA <=> NAND Flash Module device under test

The NAND Flash Controller was developed using an overview of the 2006-2007 clinic team controller.[12] After reviewing their code, it was determined that the code needed to be rewritten. The controller was developed in Verilog and interfaces to the flash device through the

daughter card provided by Micron.

The controller was created using a state machine that had three main states. The first state TOP_IDLE would wait for the command from the firmware. Once the command was received, the state machine went into TOP_COMMAND to execute each command. Once the command was executed, TOP_DONE reset all the values used to execute the command.

Timing diagrams from the Micron datasheets were used to create the Verilog for each command [3]. Each timing diagram was studied in detail and then used to create a state machine for that command. These state machines were then inserted into the TOP_COMMAND state. Figure 1 shows the timing diagram for the Reset command. An important consideration when writing these state machines was the timing issues. Some commands needed longer wait times then others. In figure 1 there were two wait times tWB and tRST that had to be accounted for so a time counter was implemented for these wait times.

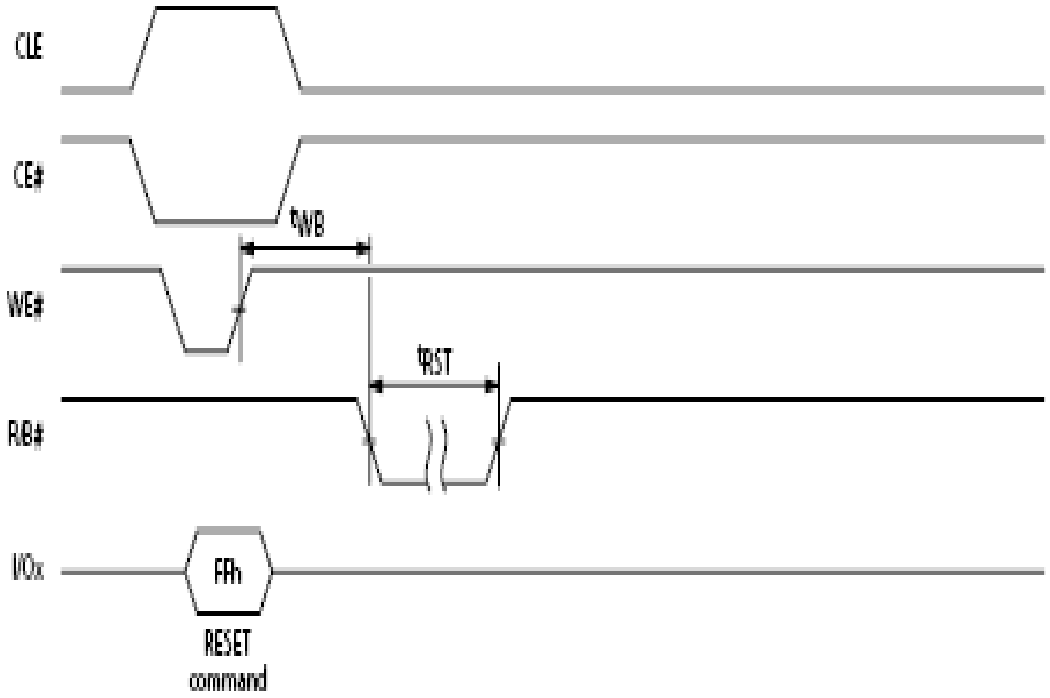

Figure 1. Reset command timing diagram

TESTING AND INTEGRATION

Considerable time was required for testing and integration, and we attempted to minimize problems in the beginning stages of development. For this reason the project was segmented to allow for individual component testing, debugging, and interfacing before it was integrated with other components. Each development task was completed by two or more team members to ensure a knowledge overlap on critical sections. Also, one team member was responsible for guiding the overall development and testing of each component.

After the individual components were completed, the development moved on to debugging to make sure that the system would endure extended testing sessions that could last upwards of 3 months time. We initially started with short test sessions over small sections of memory to make sure that all the data acquisition and analysis was correct. Much of the first stages of debugging began with simulation. Figure 2 shows an example of the simulation that was run on the state machine in the NAND Controller.

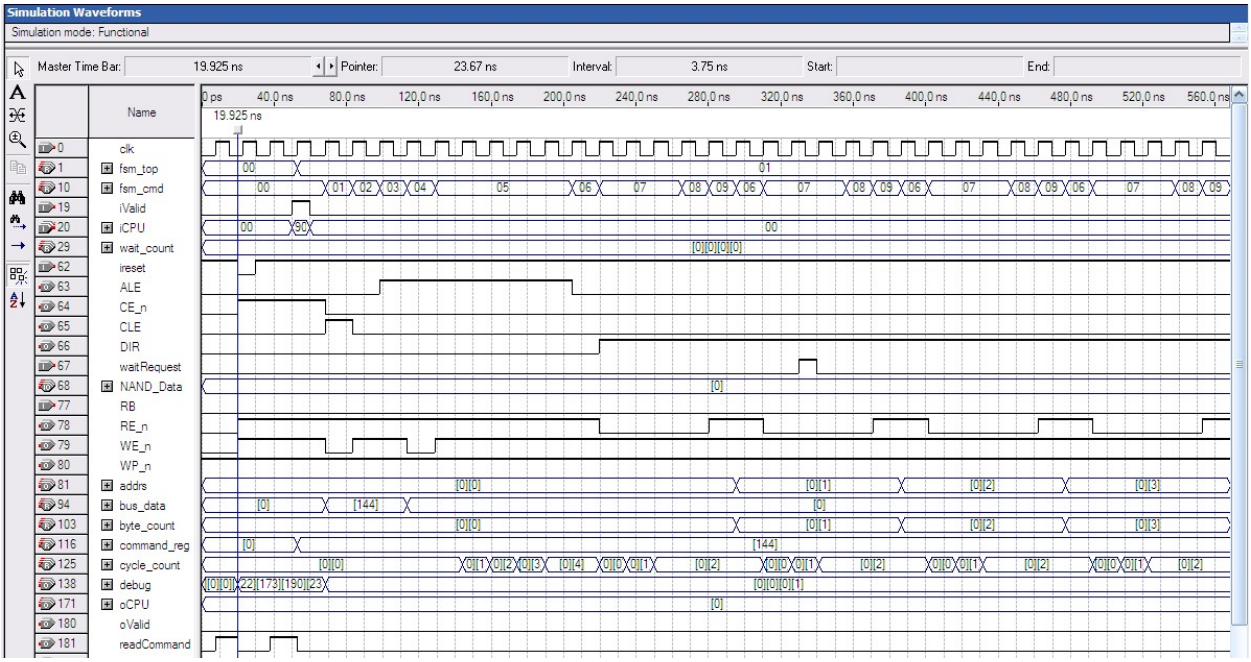

Figure 2 – Simulation using ModelSim

Simulation was the fastest way to debug major problems in the system because all of the input and output pins could all be seen at one time. Once that stage of debugging was completed, the system was in a favorable state to begin debugging on the hardware level. Some of the spare pins on the Altera development board were used to map debugging information which was then used to determine if the proper information was being transmitted and received. Not all pins could be seen at one time because the digital scope only had sixteen channels which slowed down the debugging process.

GROUP MANAGEMENT AND COMMUNICATION

Communication was essential to completing and delivering the NAND Flash test platform, but communication with future developers that will become the eventual end users of our system was also considered. A Google Code website was set up to store all the documents and meeting notes during the project development and was the central repository for all of the project source code [4]. The website also provided a permanent public presence on the Internet that can be used for ongoing issue tracking and collaboration between users.

During the project development, weekly meetings were held where all members reported on the progress of their assignments. A meeting log containing a detailed report of what was accomplished was also posted on the project website. In this manner everyone was accountable for completing the project on time. To further provide communication, each member was assigned a major role in the group and was responsible for that aspect of the project.

SCHEDULE AND MILESTONES

Task	Start Date	End Date
Roles defined and assignments made	September 18, 2007	September 25, 2007
Become familiar with project code and FPGA Project proposal and abstract	October 21, 2007	October 31, 2007
USB tested and working	November 1, 2007	November 15, 2007
FPGA communicating with host Commands sent.	November 16, 2007	December 31, 2007
New NAND Controller written. Test and debug firmware and hardware	January and February, 2008	
Present project at ECE Technical Open House Prepare final project documentation Prepare final project deliverables	March, 2008	
Collect and analyze data Submit Abstract to Flash Summit	April and June 2008	
Present project findings at Flash Summit	August 2008	

Table 2: Schedule and milestones

CONCLUSION

The integration of NAND Flash memory into new devices can lead to many improvements and technological innovations. Because of the physical limitations on read and write cycles and the lack of empirical data on failure rates, NAND Flash memory is often not utilized to its full potential. This paper describes how this test platform was created and implemented using different techniques that enhanced the production of the test platform. This test platform will allow users to ensure that NAND Flash is a reliable memory system for their specific use-scenario and will provide failure data beyond the manufacture's stated 100,000 cycle limit. The test platform will also allow users to compare different NAND Flash chips or vendors to find the product offering that best fits their application's needs.

APPENDIX

BILL OF MATERIALS

NAND FLASH VENDOR INFORMATION

I **Micron Technology** contact: Dennis Z.
8000 S Federal Way, Boise, ID 83706 F: 651-994-8186 P: 208-994-8200
Part # : 29F2G08AAC 2 Gbit w/ 8 bit I/O
Time: 3-5 Days Price/Unit Cost: $0.00
Quantity: 6 Total Cost: $0.00

II **Luscombe Engineering Company Inc.**
6465 S 3000 E Suite 101, SLC, UT 84121 P: 801-944-1280
Part # : JS29FO4G08AANB1 2 Gbit w/ 8 bit I/O
Time: 3-5 Days Price/Unit Cost: $
Quantity: 1 Total Cost: $

FPGA BOARD VENDOR INFORMATION

I **Altera** contact: http://university.altera.com/materials/boards/unv-de2-board.html
357 S McCaslin Blvd, Louisville, CO 80027 F: 303-926-4945 P: 303-926-4955
Part # : DE2 DE2 Development & Education Board
Time: 3-5 Days Price/Unit Cost: $269.00 (Academic use only)
Quantity: 1 Total Cost: $269.00 (Academic use only)

II Arches Computing contact: www.de2.archescomputing.com
Unit 708 - 222 Spadina Ave.
Toronto, Ontario, Canada M5T 3A2
Part # : DE2 DE2 Development & Education Board
Time: 3-5 Days Price/Unit Cost: $495.00 (Commercial use)
Quantity: 1 Total Cost: $495.00

USB 2.0 VENDOR INFORMATION

I **L-com, Inc.** contact: www.l-com.com/home.aspx
45 Beechwood Drive, North Andover, MA 01845 P: 978-682-6936
Part # : CSMUAA-1M Premium USB Type A-A Cable, 1.0m
Time: 3-5 Days Price/Unit Cost: $8.45
Quantity: 1 Total Cost: $8.45

II **Cables To Go** contact: www.cablestogo.com
1501 Webster Street, Dayton, OH 45404 F: 800-331-2841 P: 937-224-8646
Part # : 39979 JETLAN USB 2.0 Net/Data Transfer Cable
Time: 3-5 Days Price/Unit Cost: $22.99
Quantity: 1 Total Cost: $22.99

VENDOR LISTS

The following hardware and software components will be used to complete this project:

NAND Flash Daughter board – Provided by Micron

NAND Flash SLC chips – Provided by Micron [3]

Altera DE2 FPGA development board – Provided through the University of Utah [6]

TortoiseSVN 1.4.5 subversion client – Free Download [7]

Subclipse plugin for Eclipse/NIOS IDE – Free Download [8]

Altera Quartus II Web Edition Verilog development environment – Free Download [9]

Altera Nios II Embedded Design Suite – Free Download [9]

LibUSBDotNet – C# LibUSB-Win32 wrapper for generic USB drivers – Free Download [5]

Visual Studio 2005 - IDE for C# GUI development – VS 2005 Express download [10]

Microsoft SQL 2005 Server - SQL database engine to store results -- MS SQL Express [11]

ASSISTANCE AND ACKNOWLEDGEMENTS

Throughout the development of the project we received assistance from our faculty advisor and our project sponsor. Our faculty advisor was Ken Stevens, who helped us obtain software licenses, revise presentations and proposals, stick to project deadlines, and relay information to the project sponsor. Our project sponsor was the Micron Foundation and we received information from Micron employees Dennis Zattiero, Tim Hollis, and Dean Klein. Micron also provided the DE2 development board and the TSOP daughter hard and approved the initial project proposal. The daughter card was built by Boise State University, who retains all of the intellectual property rights associated with it.

In addition, this project was an extension of work done by a previous team and while the majority of the system has been rebuilt from scratch we have reused some of their designs and materials. The 2006-2007 Team consisted of Jeremy Hamblin, Robert Wells, David Chu, Roger White and they had the same faculty advisor and clinic sponsor. They worked on designing a similar system but were unable to complete the task due to a variety of issues. At the beginning of the project we evaluated the work that they had completed and the issues that prevented them from finishing the project, and we determined that new system architecture and a new USB driver were required. The project hardware, firmware and NAND Flash controller were redesigned to work with the SOPC builder architecture and LibUSB driver, but some of their initial research and designs are still incorporated into the final system.

Ethical Issues

Because ethics are often not considered until the time to act has passed, an effort must be made upfront to allow these issues to be dealt with properly. This includes acknowledging prior work that has been adopted or utilized, such as our incorporation of designs and materials from the 2006-2007 Micron Clinic Group. Omitting their involvement would be considered unethical. Boise State University also contributed through the development of the daughter card, and unauthorized use of their intellectual property would also be unethical. Therefore the proper acknowledgements have been made in all technical documents and public project materials. Proper IP licensing and use must also be considered with each of our development tools. Because the licenses used for Nios II, Quartus II, and ModelSim are all free web licenses, abiding by their license agreement terms is vital. Licensing terms also played an important part when choosing to use the open source libusb-win32 library to avoided the restricted terms of proprietary USB drivers regarding cost, time limitations, and redistribution.

Societal Issues

The NAND Flash test application is meant to expand the research capabilities of engineers using NAND Flash technology. Due to the fact that this project is open source, it should give a wide range of developers access to an affordable and practical testing tool that would otherwise not be available. As a result of this improved accessibility, NAND Flash development and research can become more productive, and the rate at which NAND Flash technology is used will likely increase. This will impact society through an influx of better consumer products, including portable devices and solid state hard drives. This project also acts as an educational platform for both industry and the academic world by expanding the available research on the failure characteristics of NAND Flash technology.

REFERENCES

[1] Micron NAND Flash 101 Technical Notes - Figure 1, http://download.micron.com/pdf/technotes/nand/tn2919.pdf

[2] NAND Flash Design and Use Considerations, http://download.micron.com/pdf/technotes/nand/tn2917.pdf

[3] http://2007-uofu-micron-clinic.googlecode.com/files/2_4_8gb_nand_m49a.pdf

[4] Google Code Website http://code.google.com/p/2007-uofu-micron-clinic/

[5] LibUsbDotNet, http://www.picmicrochip.com/index.php?page=LibUsbDotNet_Project

[6] Altera DE2 Board, http://www.altera.com/education/univ/materials/boards/unv-de2-board.html

[7] TortoiseSVN Client, http://tortoisesvn.net/downloads

[8] Eclipse Subversion plugin, http://www-128.ibm.com/developerworks/opensource/library/os-ecl-subversion/

[9] Altera Software https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp

[10] Visual Studio Express, http://msdn.microsoft.com/vstudio/express/

[11] MS SQL Express, http://msdn2.microsoft.com/en-us/express/aa718378.aspx

[12] NAND Flash failure behavior beyond LSC Specifications http://www.eng.utah.edu/~jhamblin/4900/index4900.html