# MODELING THE PHYSICAL CHARACTERISTICS OF NAND FLASH MEMORY

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science

Computer Science

by

**Vidyabhushan Mohan**

May 2010

© Copyright May 2010

Vidyabhushan Mohan

All rights reserved

# Approvals

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Master of Science

Computer Science

______________________________

Vidyabhushan Mohan

Approved:

| ______________________________ | ______________________________ |
|---|---|
| Sudhanva Gurumurthi (Advisor) | Kevin Skadron (Chair) |

______________________________

Mircea R. Stan

Accepted by the School of Engineering and Applied Science:

______________________________

James H. Aylor (Dean)

May 2010

# Abstract

Flash memory has gained tremendous popularity in recent years. A variant of flash, referred to as NAND flash, is widely used in consumer electronics products, such as cell-phones and music players while NAND flash based Solid-State Disks (SSDs) are increasingly displacing hard disk drives as the primary storage device in laptops, desktops, and even data centers. Computer architects have recently begun exploring the use NAND flash, from SSD organizations to disk caches and even new flash-based server architectures. In order to study this design space, architects require simulation tools that can provide detailed insights into the behavior of flash memory. This thesis presents two such tools that model two important characteristics of NAND flash: power consumption and endurance.

The first tool, called *FlashPower*, is a microarchitecture level modeling tool that provides a detailed analytical power model for Single-Level Cell (SLC) based NAND flash memory. *FlashPower* estimates the power consumed by a NAND flash memory chip during its various operating modes. We have validated *FlashPower* against published chip power measurements and show that they are comparable. Results from a design space exploration using *FlashPower* indicate that the values of bits being read or written into NAND flash memory have a significant impact on energy dissipation.

The second tool, called *Flash EnduraNCE (FENCE)*, models the endurance characteristics of NAND flash and captures the impact of stress and recovery on NAND flash memory cells. Using *FENCE*, we show that the recovery process, which prior studies on flash based SSDs have not considered, significantly boosts endurance. Using a set of real enterprise workloads, we show that this recovery process allows for orders of magnitude higher endurance than those given in datasheets. Our results indicate that, under realistic usage conditions, SSDs that use standard wear-

leveling techniques are more resilient than previously assumed and support a wider range of I/O intensive enterprise applications.

# Acknowledgments

This thesis would not have been possible without the help of so many people. Of all of them, I would like to first thank my advisor Professor Sudhanva Gurumurthi. Sudhanva's constant support and timely guidance have been instrumental in helping me realize what I am capable of. He was there to listen and correct me whenever I made mistakes. His advice on how to communicate research ideas, both in writing and during a discussion, has helped me a lot in improving my communication skills. His financial support has ensured that I focus fully on my work without having to worry about my livelihood. Along with Sudhanva, I would like to thank my co-advisor Professor Mircea R. Stan. Discussions with Mircea have helped me understand the circuit-level concepts and his insightful questions have helped me to improve the quality of my thesis and broaden the scope of my thinking. Weekly meetings involving Sudhanva, Mircea and myself have been an absolute delight and I hope to have many such stimulating discussions in the future. I also take this opportunity to thank my committee chair, Professor Kevin Skadron.

I would like to thank my family for their unwavering faith in my abilities. Their confidence in me is more than what I have in myself. They have encouraged and backed the important decisions that I have made and the knowledge of their backing has lifted me from my low spirits and kept me going. I draw a lot of inspiration from my mom and dad, who despite several hardships, have made sure that their children get the best in everything that they desire. Mom and dad, I dedicate this thesis to you. I would like to thank my sister, Apoorva for her support and confidence in me. I would also like to thank all my relatives for their support.

I am fortunate to have an excellent group of friends around me. They have entertained me, kept me on my toes and during difficult times, have provided a shoulder to lean on. Thanks guys. Special

thanks to Ram, whose guidance has been instrumental in helping me improve my decision making abilities. Being far away from one's home country, Ram's presence has made me feel at home, here in Charlottesville.

Finally, I bow under the almighty and thank HIM for everything. *Krishnarpanam asthu.*

# Contents

**1 Introduction** **1**

**2 NAND Flash Memory Primer** **4**

2.1 The NAND Flash Memory Array . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.2 The Floating Gate Transistor (FGT) . . . . . . . . . . . . . . . . . . . . . . . . 8

2.3 The Read Operation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.4 The Program operation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.5 The Erase Operation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.6 Single-Level Cells (SLCs) vs Multi-Level Cells (MLCs) . . . . . . . . . . . . . . 13

**3 *FlashPower*: A Detailed Analytical Power Model** **14**

3.1 Circuit Components . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3.2 Power State Machine . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.3 Integration with CACTI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.4 Power Modeling Methodology . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

**4 Validation of *FlashPower*** **28**

**5 *FENCE*: An Endurance Model for NAND Flash** **34**

5.1 FENCE: An Architecture-Level Model for Estimating Endurance . . . . . . . . . . 35

5.2 Impact of Charge Detrapping on SSD Endurance . . . . . . . . . . . . . . . . . . 39

**6 Analysis of SSD-Level Endurance** **44**

6.1 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

**7 Related Work** **49**

**8 Conclusions and Future Work** **51**

**Bibliography** **53**

# List of Figures


2.1 A NAND Flash Memory Chip . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 A NAND Flash Memory Array . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.3 A Floating Gate Transistor . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.1 Power State Machine for a SLC NAND Flash Chip . . . . . . . . . . . . . . . . . . 15

3.2 Modeling a Floating Gate Transistor . . . . . . . . . . . . . . . . . . . . . . . . 16

4.1 Validation Results for Read Operation . . . . . . . . . . . . . . . . . . . . . . . 30

4.2 Validation Results for Program Operation . . . . . . . . . . . . . . . . . . . . . 31

4.3 Validation Results for Erase Operation . . . . . . . . . . . . . . . . . . . . . . 32

5.1 Threshold voltage distribution for a 2-bit MLC . . . . . . . . . . . . . . . . . . . 40

5.2 Increase in $\delta V_{th}$ with P/E cycles for different recovery periods. . . . . . . . . . . . 41

6.1 Endurance results for enterprise workloads. . . . . . . . . . . . . . . . . . . . . . 47

# List of Tables


3.1 Inputs to *FlashPower*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

4.1 Microarchitectural parameters of the validation chips. . . . . . . . . . . . . . . . . 28

5.1 Endurance limits with charge detrapping. . . . . . . . . . . . . . . . . . . . . . . . 42

6.1 SSD Configuration Used for Evaluation. . . . . . . . . . . . . . . . . . . . . . . . . 44

6.2 Recovery time distributions. $[X, y)$ indicates recovery periods of duration $t$ where $X \leq t < y$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48

# Chapter 1

# Introduction

Flash memory is the most popular solid-state memory used today. Although initially used only in consumer electronics, such as cellphones and portable music players, the drop in the price of NAND flash memory has paved the way for its use in mass storage devices as well, in the form of Solid State Disks (SSDs). SSDs are replacing Hard Disk Drives (HDDs) as the storage of choice in laptops, desktops and even servers. While most research on flash has focused on the device technology and the circuit-level design of flash chips [7], or on high-level system issues such as file-system and Flash-Translation Layer design [12], there has been growing interest in the computer architecture community on flash memory. Computer architects have begun exploring a variety of topics related to flash, including the design of SSDs [9], disk-caches [21], new flash-based server architectures [3] and even hybrid memories [23]. In order to study this rich architecture design space and support such a wide range of systems, architects require simulation tools that can provide detailed insights into the characteristics of NAND flash.

Two important characteristics of NAND flash that influence architecture are power consumption and endurance. Power is an important consideration because the design of a NAND flash based memory array is closely related to the power budget within which it is allowed to operate. For example, NAND flash used in consumer electronic devices has a significantly lower power budget compared to that of a SSD used in data centers, while the power budget for NAND flash based disk caches is between the two. Therefore, we require tools that can accurately estimate the power consumption of various memory organizations and tailor the design of NAND flash to the power

budget. In order to achieve this, the thesis presents *FlashPower*, a detailed analytical power model for Single-Level Cell (SLC) NAND flash based memory chips, which are used in high-performance flash based architectures. *FlashPower* models the key components of a flash chip during the read, program, and erase operations and when idle and is parameterized to facilitate the exploration of a wide spectrum of flash memory organizations. Results from this design space exploration using *FlashPower* indicate that the values of bits being read or written into NAND flash memory have a significant impact on the energy dissipated by NAND flash, providing an opportunity to perform architecture level power optimizations such as invert coding [44].

Endurance is another property of NAND flash memory that is of interest to architects and system designers. Even though SSDs offer high performance, low power, and greater ruggedness compared to HDDs, one of the main impediments to the wide adoption of SSDs in servers has been its apparent limited endurance. Flash memory blocks can wear out after a certain number of write (program) and erase operations. Manufacturer datasheets quote values that range from 10,000-100,000 program/erase (P/E) cycles for NAND flash endurance. Architecture and systems papers that explore the use of flash memory use these values to estimate endurance [2, 21, 34, 37]. However, at the physical level, endurance is a more complex process, involving stresses due to charge trapping in the tunnel oxide of the floating gate transistors induced by P/E operations, but also a recovery process that detraps the charge and partially heals the devices [51, 30]. By modeling both stress and recovery, we can get deeper insights into the endurance characteristics of flash and make a more accurate assessment of SSD endurance in enterprise storage systems. Recent papers on NAND flash chip measurements provide evidence that endurance is higher than the values reported in datasheets [6, 14], which further motivates studying this phenomenon in more detail.

To summarize, the thesis makes the following contributions:

- We develop *FlashPower*, a highly parameterized and detailed analytical model that estimates the power consumption of a SLC based NAND flash memory chip during its various operating modes.

- We validate *FlashPower* against published chip power measurements [14] and show that the

power estimations provided by our tool are comparable to the measured values.

- We develop *Flash EnduraNCE (FENCE)*, an analytical endurance model for NAND flash memory suitable for architecture level simulations.

- We use *FENCE* to quantify the impact of charge trapping and detrapping for both Single-Level Cell (SLC) and Multi-Level Cell (MLC) NAND flash memory.

- We study the endurance of enterprise-class SSDs when exercised by several real data center workloads. We show that, due to charge detrapping, NAND flash that uses standard wear-leveling techniques can support orders of magnitude higher number of P/E cycles than those given in datasheets. These results indicate that SSDs can be safely deployed in data centers without endurance concerns and can support a much wider range of applications than previously assumed.

The organization of the thesis is as follows: Chapter 2 provides a detailed overview of NAND flash memory, its microarchitecture and operation. We then present *FlashPower* in Chapter 3 followed by validation of *FlashPower* in Chapter 4. Chapter 5 describes *FENCE* and analyzes the impact of charge detrapping on flash endurance at a transistor level. Chapter 6 provides a detailed analysis of SSD-level endurance using *FENCE*. Chapter 7 discusses the related work and Chapter 8 presents the future work and concludes the thesis.

# Chapter 2

# NAND Flash Memory Primer

Flash is a type of EEPROM (Electrically Erasable Programmable Read-Only Memory) that supports read, program and erase as the basic operations. As shown in Figure 2.1, a Flash Memory chip includes a command and a status register, a control unit, a set of decoders, some analog circuitry, buffers, address and data buses, and the Flash Memory array. The address or data buses also carry command signals to the memory chip. An external memory controller sends a read, program, or erase command to the Flash Memory chip along with the appropriate physical address. For a read operation, the controller passes the physical address to the chip which in turn transfers the data back to the controller. For a program operation, the data and physical address to be programmed is passed through the data and address bus to the chip. For an erase operation, only the physical address is passed to the chip. Depending on the type of operation to be performed, the control unit within the chip enables the decoders which use the address bits to select the appropriate physical block. The control unit is also responsible for activating the correct analog circuitry to generate high voltages needed for program and erase operations.

The Flash Memory array is a two-dimensional array of semiconductor memory similar in structure to those used in other types of memories such as SRAMs and DRAMs. The array is a matrix like structure composed of rows (connected to word-lines) and columns (connected to bit-lines). At the intersection of a row and a column is a Floating Gate Transistor (FGT) which stores logical data. In this thesis, the term memory cell and the term FGT refer to the same physical entity and

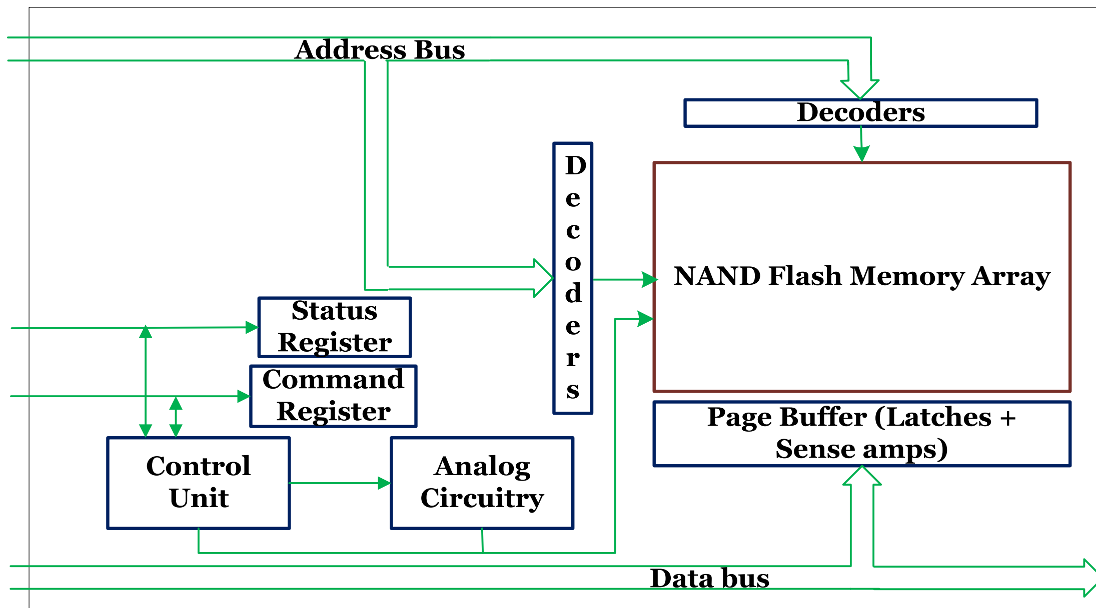



Figure 2.1: A NAND Flash Memory Chip

are used interchangeably. These memory cells that store one bit of data per cell are referred to as Single-Level Cells (SLCs), while memory cells that store multiple bits of data per cell are referred to as Multi-Level Cells (MLCs). The organization of FGTs inside the array determine whether the Flash Memory is a NAND or NOR memory.

For NOR flash, FGTs are connected in parallel to a bit-line with the control gates connected to a word-line while for NAND flash, the FGTs are connected in series (see Figure 2.2). The two ends of this serial connection are connected to access transistors which in turn are connected to the bit-line. So NAND flash has fewer contacts than NOR flash and resulting in higher cell densities. The NAND cells can be programmed faster than NOR but their read latency is higher compared to NOR. Another difference between them is that the NAND memory is *not* bit-programmable while the NOR memory *is* bit programmable. Given the inherent characteristics of these devices, NAND memory is the memory of choice for storing large amounts of data and is hence popular for use in SSDs and hybrid memory storage, while NOR memory is the memory of choice in ROMs, small controllers which are seldom written but require fast read access time. As our goal is to develop Flash models that are useful for exploring the design of mass storage devices like SSDs

and extensions to the memory hierarchy, this paper focuses only on NAND Flash memory.

Finally, the flash memory chip comprises of latches and sense amplifiers, which constitute the buffers (referred to as page buffers). The latches store the data that is transferred to/from the memory array while the sense amplifiers sense the bit-lines during a read operation. The status register is used by the controller to monitor the status of the command that was sent to the chip. The controller also includes ECC (Error Checking and Correction) logic to address failure and reliability issues encountered in the chip and to ensure that correct data is written to or read from the chip.

## 2.1 The NAND Flash Memory Array

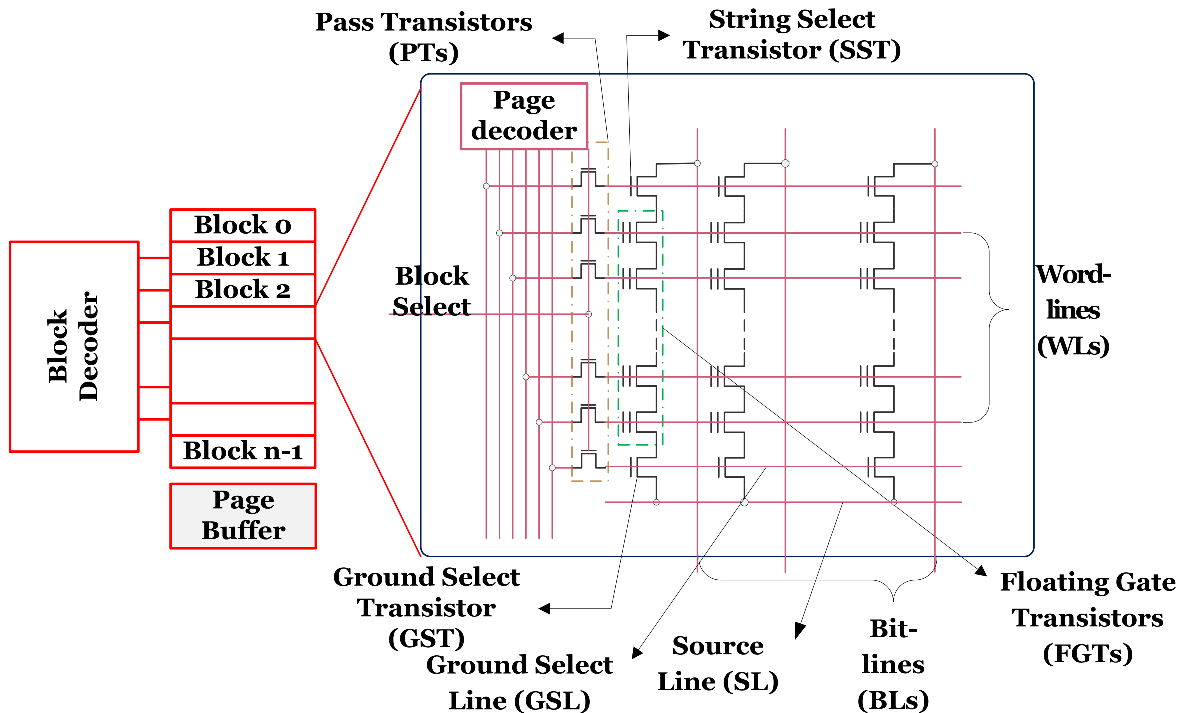



Figure 2.2: A NAND Flash Memory Array. Adapted from [7].

Figure 2.2 shows the block diagram for the NAND memory array. The NAND flash memory array is partitioned into *blocks* that are, in turn sub-divided into *pages*. A page is the smallest granularity of data that can be addressed by the external controller. Some devices like [39] allow

sub-page accesses by the controller. A set of FGTs connected in series is referred to as a *string*. The number of FGTs in a string is equal to the number of pages in a block. In figure 2.2, each column corresponds to a string while each row corresponds to a page. Word-lines select a page of memory to perform read or program operation. These operations first involve selecting a block using a block decoder following which one of the rows in a block is selected using a page decoder. Since NAND memory does not support in-place updates, a page needs to be erased before its contents can be programmed; but unlike a program or a read operation which work at a page granularity, the erase operation is performed at a block granularity. The reason for this is explained in the next subsection. The drain of the String Select Transistor (SST) connected to each bit-line controls the bit-line biasing to each string while its gate is connected to the String Select Line (SSL) that switches the SST on and off. A pass transistor connected to each word-line controls the word-line biasing for each row in the array. A Ground Select Transistor (GST) connects the other end of the string to the Source Line (SL) which connects the source of one-end of the FGT to the power supply based on the type of operation to be performed.

The size of each page (in bits) corresponds to the number of FGTs in each row of the array. However, the number of FGTs in each row is higher than the advertised capacity of each page. This is because each page contains a set of spare FGTs in a spare area along with the FGTs in the data area. The purpose of this spare area is to store the ECC bits corresponding to the data bits of that page as well as a physical-to-logical address mapping (like an inverted page table). In general, the controller may choose to save additional metadata information about the page in the spare area. During a read operation, the entire page (including the bits in the spare area) is transmitted to the controller. The ECC logic in the controller uses this information to check for the correctness of the read data.

During a page program operation, the controller transmits both the actual data and the ECC bits to the Flash memory. Upon system boot, the controller also scans the spare area of each page in the entire memory array to load the logical to physical address mapping into its own memory. The controller uses a Flash Translation Layer (FTL) to determine the logical-to-physical address mapping. In addition to mapping, the FTL also performs garbage collection to erase stale copies

of pages that are left behind on flash due to the non in-place writes and performs wear-leveling operations to ensure that all the flash blocks wear evenly. More information about FTLs can be found in [12, 5, 15].

## 2.2 The Floating Gate Transistor (FGT)

Figure 2.3 sketches a basic FGT. The FGT is similar to a NMOS transistor except for an additional floating gate between the channel and the control gates. The floating gate is surrounded by an insulator that helps in the retention of charge (typically electrons) in the floating gate; this retention is long enough (in the order of tens of years) for the memory to be non-volatile. Hence using a combination of the floating gate (to store charges) and the insulator (to trap charges), a FGT provides non-volatility. The presence or absence of charges inside the floating gate causes a shift in the threshold voltage of the FGT which is used to distinguish a logical "1" from a logical "0".

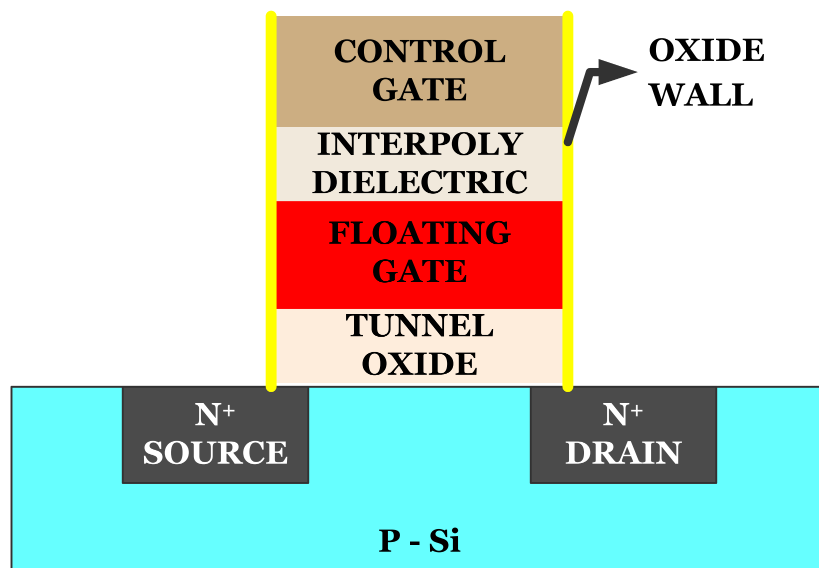



Figure 2.3: A Floating Gate Transistor

A read operation involves detecting whether the floating gate is charged or not, while Fowler-Nordheim (FN) tunneling [27] is used to tunnel charges to and from the P-well to the floating gate. Tunneling of charges to the floating gate from the P-well corresponds to the program operation

while tunneling in the other direction corresponds to the erase operation. Some examples in the literature use logical “0” as the presence of charge in the floating gate [7], while others [46] use logical “1” to indicate the presence of charge in the floating gate. In this thesis, we assume that cells corresponding to logical “0” contain charge in the floating gate.

Due to the presence of an insulator between the channel and the floating gate, a high electric field (provided by a high voltage) is necessary to enforce FN Tunneling [27]. This means that tunneling is a high power consuming operation. The same is the case for removing charges from the floating gate. Because of this, a single FGT is never programmed alone. Only a group of FGTs is programmed at a time; this group of FGTs correspond to a *page* (as shown in Figure 2.2).

While this explains why NAND is block addressed instead of bit addressed, another feature of NAND flash memory is that it does not support in-place update operations. An in-place update operation involves removing charge from an already charged floating gate or vice versa. Tunneling (in either direction) involves high voltage and can cause disturbance to nearby FGTs. This disturbance can be overcome using what are known as inhibit modes [45] but this ends up consuming high power because other nearby FGTs need to be biased to high voltages to prevent them from tunneling. As a result both tunneling and inhibition of tunneling are expensive operations and place considerable stress on a FGT. To reduce the impact of the stress on a FGT due to tunneling in both directions, program operations can only tunnel electrons in one direction (from the channel to the floating gate) in NAND flash. Removing charges from the floating gate is performed at a granularity of a block as in Figure 2.2. The blocks are physically laid out so that they share a single P-well, and biasing the P-well at a very high voltage causes tunneling off the floating gate. Since all the FGTs in the block are tunneled simultaneously, the stress on FGTs is considerably reduced. This operation is referred to as an erase operation. Now we explain how the read, program and erase operations are performed on the NAND flash memory array.

## 2.3 The Read Operation

To perform a read operation, the external controller determines the physical address of the page to be read based on its logical-to-physical address mapping. This address is then passed to the control unit in the chip, which selects the correct page using the set of block and page decoders. Once a block is selected, current sensing of bit-lines is employed to determine whether the floating gate in a NAND memory cell contains charge or not. This is achieved by driving the control gate voltage of the un-selected word-lines to $V_{read}$. This causes other memory cells in the string to serve as transfer gates. The control gate voltage of the selected word-line is set to 0V. If the floating gate does not contain charge, then the memory cell is on (transfer is depletion type) and current flows through the string. If the floating gate is charged, then the memory cell is off (transfer is enhancement type) because the floating gate has a higher threshold voltage and current does not flow through the string. Sense amplifiers connected to one end of the bit-line sense the flow of current. The absence of current flow is considered to be a logical "0" while its presence indicates a "1". The data is then transferred from the latches to the controller where the ECC logic performs error checking and correction.

## 2.4 The Program operation

To perform a program operation, the external memory controller needs to determine the physical address of the page to be programmed. For each write operation, a free valid page needs to be chosen because NAND flash does not allow in-place update operation. FTL algorithms are used to determine a free valid page that can be used for programming. Once such a page is chosen, the controller updates its internal data structure to indicate that the logical address now corresponds to a new physical page. The previous physical page is marked as invalid. If the number of free but valid pages becomes low, the controller has to reclaim invalid pages. This is performed using an erase operation, which is explained in the next sub-section. Once the invalid pages are reclaimed, the controller can use them for future programming operations. The controller then transmits the program command, the data to be programmed and the physical address of the page to the chip.

When a request for a program operation arrives from the controller, a row of the memory array (corresponding to the requested page) is selected and the latches in the page buffer are loaded with the data to be written. The SST is then turned on while the GST is turned off by the control unit. For FN tunneling to occur, a high electric field is necessary across the floating gate and the substrate. This high electric field is achieved by setting the control gate of the selected row to a high voltage $V_{pgm}$, and biasing the bit-lines corresponding to logical "0" to ground. This creates a high potential difference across the floating gate and the substrate causing electrons to tunnel from the substrate onto the floating gate. For "1" programming (which is basically non-programming), the memory cell should remain in the same state as before the program operation. While different techniques are adopted to prevent tunneling of electrons for such cells, we assume the self-boosted program inhibit operation described in [45]. This technique provides the necessary program inhibit voltage by driving the bit-lines corresponding to logical "1" to $V_{cc}$ and by turning on the SSL and turning off the GSL. When the word-line of the selected row rises to $V_{pgm}$ , the series capacitance through the control gate, floating gate, channel and the bulk are coupled, boosting the channel potential automatically and preventing FN tunneling. More details of the self-boosted program inhibit scheme are given in [7].

An important aspect of programming is a program-verify operation which checks whether the memory cell has been successfully written or not. This extra step is necessary because a read operation of a cell is dependent on the threshold voltage of other cells that are connected in series in the string and hence it is necessary to maintain the threshold voltage of each cell [22]. Otherwise, a read operation on one cell would be affected by other cells in the string which have been programmed with a different threshold voltage. To ensure that each cell has been programmed with the correct threshold voltage, the write operation is performed in small increments, and after each write step, a read operation checks if sufficient charge has been deposited onto the floating gate. Transferring charge to the floating gate in small increments also increases the reliability of the cells by restricting the exposure of the floating gate to tunneling to short time periods. This is important because the write operation involves high voltage (of the order of 10V-20V) to effect tunneling, and continued exposure can weaken the tunnel oxide which can result in faulty cells [4]. This means

that the controller issues a single program command to the memory array while internally the program operation is performed along with a program-verify operation to correctly program a cell. The program and verify-read operations are typically performed multiple times before reporting error to the controller [46]. In the case of an error, the controller marks the page as invalid and copies the data to another free but valid page.

Narrow word-line pitch and high voltage in selected word-lines may disturb other word-lines in the same block. To prevent the disturbance from affecting the contents of the cells in other word-lines, the other word-lines are biased at a pass voltage $V_{pass}$. While this shields the word-lines in the same block from being disturbed, other nearby un-selected blocks may also be disturbed. This is prevented by biasing their pass and select transistors to ground.

## 2.5 The Erase Operation

When the FTL in the controller determines that the number of free pages is critically low, the controller sends an erase command to the chip. The FTL layers have their own policy to determine which block should be reclaimed. Once the physical address of the block to be erased is determined by the controller, it is passed to the chip. The control unit in the chip selects the block to be erased using the block decoder.

The control gates of all the word-lines in the selected block are grounded and the P-well of the block is biased at $V_{erase}$. This causes FN tunneling of charges off the floating gate to the P-well for 0-programmed cells while the un-programmed cells corresponding to “logical 1” are over-erased. Unlike the program operation where tunneling happens across multiple program pulses, erasure happens in a single erase pulse because over-erasure is not a concern in NAND memory [7]. Since multiple blocks in an array share a common P-well [7], it is possible that unselected blocks in the same array sharing the same P-well may inadvertently be erased. Different techniques [45] have been proposed to inhibit unselected blocks from being erased. One such technique is called self-boosted erase inhibit [45] which utilizes the capacitive coupling between the control gates and the P-well. Before applying the erase voltage to the P-well, the control gates of the un-selected

blocks that share the same P-well with the selected block are floated by shutting off the pass gates. The control gates are coupled up as the P-well is biased to $V_{erase}$ which causes the unselected blocks to be automatically inhibited from being erased. More details about this scheme are given in [45]. This scheme is used in latest generation devices manufactured by Samsung, Toshiba and SanDisk [26, 32].

## 2.6 Single-Level Cells (SLCs) vs Multi-Level Cells (MLCs)

The operations that are explained above correspond to Single-Level Cells (SLCs) where the presence/absence of charge in the floating gate is used to represent a single bit. Since a logical bit corresponds to an analog voltage, multiple bits can be stored in a single cell by having multiple levels of voltage inside a floating gate. These are referred to as Multi-level cells (MLCs) which behave the same way as Single-level Cells (SLCs) but require complex sense amplifiers to differentiate multiple voltage levels in the floating gate. Because varying voltage levels correspond to different logical bits, the program operation should be performed slowly to make sure that no excess or less charge is tunneled into the floating gate. Both these factors result in the read and program time to be slower for MLCs compared to SLCs. Lifetime of MLCs is also considerably lesser than SLCs because MLCs quickly lose their ability to store varying levels of voltage. Hence MLCs have lower reliability compared to SLCs. Because MLCs have lower performance and lower reliability compared to SLCs, they are not suitable in performance critical and data integrity critical environments like servers. However, MLCs are cheaper alternatives to SLCs because they provide higher capacity for the same cost. Hence their use is limited to consumer electronics and personal storage.

The power and the endurance model described in this thesis use the knowledge of the microarchitecture and operation of flash memory. The power model that we describe is applicable for SLC based NAND flash memory while the endurance model is applicable irrespective of the whether the memory is SLC or MLC flash.

# Chapter 3

## *FlashPower*: A Detailed Analytical Power Model

This chapter presents the details of the analytical power model that we have developed for SLC based NAND flash memory chips. *FlashPower* models the energy dissipated during the basic flash operations and when the chip is idle. Before delving into the details of the model, we list the components that are modeled and the power state machine. We then explain how *FlashPower* was integrated with CACTI [50], followed by the details of the model itself. We then conclude this section with a discussion of the current limitations of *FlashPower*.

### 3.1 Circuit Components

With respect to Figure 2.2, the components that dissipate energy are,

- The bit-line (BL) and word-line (WL) wires.
- The SSL, GSL and SL.
- The drain, source and the gate of the SST, GST and PTs.
- The drain, source and control gate of the FGTs.
- The floating gate of the FGTs - Energy dissipated during program and erase operation.

In addition to the above components, the energy dissipated by the block and page decoders, the sense amplifiers present in the page buffer (for the read operation), the charge pumps (that provide

high voltages for program and erase operation) and the I/O pins are modeled. The energy per read, program and erase operation is determined by aggregating the energy dissipated by all the aforementioned components.

## 3.2 Power State Machine

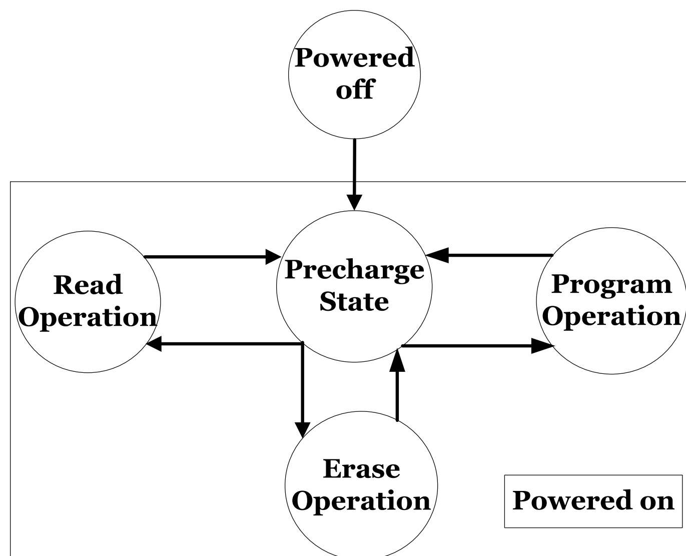



Figure 3.1: Power State Machine for a SLC NAND Flash Chip

Figure 3.1 describes the power state machine for a SLC NAND flash chip. The circles represent the individual states, while the solid lines denote state transitions. We model the energy dissipated when the chip is powered. When the chip is on but is not performing any operations, it is said to be in a "precharge state". In this state, the bit-lines are precharged while the word-lines, and the select lines (SSL, GSL and SL) are grounded. The array is isolated by the select lines but is ready to respond to commands from the controller. Upon receiving a read, program, or erase command from the controller, the state machine switches to the corresponding state. This state transition along with the actual operation dissipates energy, which we model. Upon completion of the command, the state machine switches back to the precharge state, dissipating energy in the process.

## 3.3 Integration with CACTI

*FlashPower* is designed as a plug-in to CACTI 5.3 [50], which is a widely used memory modeling tool. The power consumed by array peripherals such as decoders and sense amplifiers are estimated assuming that they are high performance devices and the bit-lines and word-lines are estimated assuming aggressive interconnect projections. We also model a FGT as a CMOS transistor and then use the CMOS transistor models of CACTI to calculate the parasitic capacitance of a FGT. However, unlike CACTI, which models individual components of a memory system, we have chosen to model the basic operations on the flash memory. This is because, in the memories that CACTI models, individual operations operate at the same supply voltage to drive the bit-lines and the word-lines. For example in a SRAM cache, the voltage at which the word-line is charged is same for both read and write. The granularity of a read/write is also the same. However for NAND flash, read, program and erase operate at different bias conditions and the granularity of an erase differs from read/program. Since the circuitry behaves differently for these different operations, we believe that it is more appropriate to model energy for the basic operations on the flash memory. The inputs to *FlashPower* are summarized in Table 3.1.

## 3.4 Power Modeling Methodology

### 3.4.1 Modeling a FGT

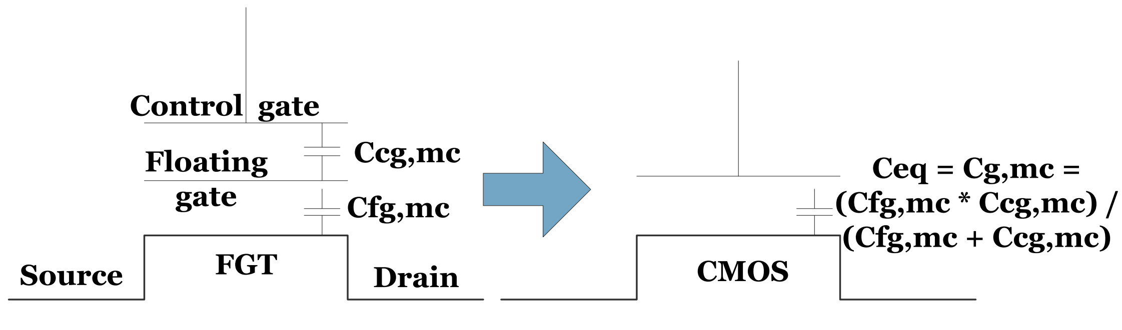



Figure 3.2: Modeling a Floating Gate Transistor

Table 3.1: Inputs to *FlashPower*.

| Microarchitectural Parameters | |
|---|---|
| $N_{pagesize}$ (R) | Size (in bytes) for the data area of each page. |
| $N_{sparebytes}$ (R) | Size (in bytes) for the spare area for each page. |
| $N_{pages}$ (R) | Number of pages per block. |
| $N_{brows}$ (R) | Number of rows of blocks in a plane. |
| $N_{bcols}$ (O) | Number of columns of blocks in a plane. |
| $N_{planes}$ (O) | Number of planes per die. |
| $N_{dies}$ (O) | Number of dies per chip. |
| $tech$ (R) | Feature size of FGTs. |
| **Device-level Parameters** | |
| $N_A, N_D$ (O) | Doping level of P-well and N-well. |
| $\beta$ (O) | Capacitive coupling between control gate and the P-well. |
| GCR (O) | Ratio of control gate to total floating gate capacitance. |
| **Bias Parameters** | |
| $V_{dd}$ (R) | Maximum operating voltage of the chip. |
| $V_{read}$ (O) | Read voltage. |
| $V_{pgm}$ (O) | Program voltage for selected page. |
| $V_{pass}$ (O) | Pass voltage during program for un-selected page. |
| $V_{era}$ (O) | Erase voltage to bias the substrate. |
| $V_{bl,pre}$ (O) | Bit-line precharge voltage. |
| **Timing Parameters** | |
| $t_{program}$ (O) | Latency to program a page after data is loaded in the page buffer. |
| $N_{loop}$ (R) | Maximum partial program cycles. |
| $t_{read}$ (R) | Latency to read a page from a plane to the page buffer. |
| **Workload Parameters** | |
| $N_{bits_1}$ (O) | Number of 1's to be read, programmed or stored. |
| **Chip-level Parameters** | |
| $C_{pin}$ (R) | Capacitance of I/O pins in the chip. |

(R) - Indicates required argument.
(O) - Indicates optional argument.

To calculate the energy dissipation, it is necessary to estimate a FGT's parasitic capacitance - i.e. a FGT's source, drain and gate capacitance. While the source and the drain of a FGT and a CMOS transistor are similar, FGT has a two gate structure (floating and control) while CMOS has a single gate. A dual gate FGT structure is modeled as a single gate CMOS structure and the parasitic capacitance of this CMOS structure is modeled using CACTI. Modeling a FGT as a CMOS structure is done by calculating the equivalent capacitance of the two capacitors (one across the inter-poly dielectric and other across the tunnel oxide). This is illustrated in Figure 3.2. In the figure, the control gate capacitance $C_{cg,mc}$ is calculated using the information on gate coupling ratio(GCR) available in [20], while the floating gate capacitance $C_{fg,mc}$ is calculated using the overlap, fringe and area capacitance of a CMOS transistor of the same feature size. The source and drain capacitance of the transistor depends on whether the transistor is folded or not. *FlashPower* assumes that the transistor is not folded as the feature size is in the order of tens of nanometers. The drain capacitance is modeled using CACTI. Other CMOS transistors like the GST, PT and SST are modeled using CACTI.

### 3.4.2 Derived Parameters

The length of each bit-line $L_{bitline}$ and each word-line $L_{wordline}$ are derived as:

$$L_{wordline} = N_{bitlines-perblock} * N_{bcols} * pitch_{bit-line} \tag{3.1}$$

$$L_{bitline} = (N_{pages} + 3) * N_{brows} * pitch_{word-line} \tag{3.2}$$

where $N_{bitlines-perblock} = (N_{pagesize} + N_{sparebytes}) * 8$. "3" is added to $N_{pages}$ because for each block, the bit-line crosses 3 select lines (GSL, SSL and SL). The terms $pitch_{bit-line}$ and $pitch_{word-line}$ refer to the bit-line and word-line pitch respectively and are equal to $2 * featuresize$.

The total capacitance to be driven along a word-line is given by

$$C_{wl} = C_{d,pt} + C_{g,mc} * N_{bitlines-perblock} + C_{wl,wire} * L_{wordline} \tag{3.3}$$

where, $C_{d,pt}$ is the drain capacitance of the pass transistor, $C_{g,mc}$ the equivalent gate capacitance of FGT and and $C_{wl,wire}$ is the word-line wire capacitance. Similarly, the total capacitance to be driven along the bit-line is equal to

$$C_{bl} = 2 * C_{d,st} + C_{d,mc} * N_{pages} + C_{bl,wire} * L_{bitline} \tag{3.4}$$

where $C_{d,st}$ is the drain capacitance of the select transistors (SST), $C_{d,mc}$ is the drain and source capacitance of the FGT and $C_{bl,wire}$ is the bit-line wire capacitance. It is assumed that adjacent FGTs connected in series share the source and the drain. Because of this assumption, the source capacitance of one FGT is considered equal to the drain capacitance of the neighboring FGT. The source of SST is shared with the drain of the first FGT in the string while the source of the last FGT in the string is shared with the drain of GST. The total capacitance of the SSL is calculated as,

$$C_{ssl} = C_{gsl} = C_{d,pt} + C_{g,st} * N_{bitlines-perblock} + C_{wl,wire} * L_{wordline} \tag{3.5}$$

It should also be noted that the GSL needs to drive the same components as the SSL and hence $C_{ssl} = C_{gsl}$. The capacitive component of the SL is equal to:

$$C_{srcline} = C_{wl,wire} * L_{wordline} + C_{d,st} \tag{3.6}$$

It is assumed that the source capacitance of GST is equal to it's drain capacitance ($C_{d,st}$).

### 3.4.3 Transition from the Idle to the Precharged State

When this transition occurs, the bit-lines and word-lines are precharged to $V_{bl,pre}$ and $V_{wl,pre}$ respectively. The SST, GST and the PTs are biased so that the current flowing through the FGTs is disabled. The source line is also grounded. The total energy dissipated by the array while in the

precharge state ($E_{pre}$) is given by:

$$E_{pre} = E_{bl,pre} + E_{wl,pre}$$

$$E_{bl,pre} = 0.5 * C_{bl,wire} * (0 - V_{bl,pre})^2 * N_{bitlines-perblock} * L_{bitline}$$

$$E_{wl,pre} = 0.5 * C_{wl,wire} * (0 - V_{wl,pre})^2 * N_{pages} * L_{wordline}$$

While bit-lines are generally precharged to reduce the latency of read accesses [26], the word-lines are not. During our validation $V_{wl,pre}$ was set to zero, but the model includes the word-line precharge $V_{wl,pre}$ as a parameter so that devices that perform word-line precharging can use this parameter.

### 3.4.4 The Read Operation

To perform a read operation, the block decoder selects one of the blocks to be read while the page decoder selects one of the $N_{pages}$ inside a block. In the selected block, the word-line of the *selected* page is grounded while the *un-selected* word-lines are biased to $V_{read}$ from $V_{wl,pre}$. This causes the un-selected pages to serve as transfer gates. Hence the energy dissipated in biasing the word-line of the selected page to ground and the un-selected pages to $V_{read}$is equal to:

$$E_{selected-page,r} = 0.5 * C_{wl} * (0 - V_{wl,pre})^2 \tag{3.7}$$

$$E_{unselected-pages,r} = 0.5 * C_{wl} * (V_{read} - V_{wl,pre})^2 * (N_{pages} - 1) \tag{3.8}$$

For the read operation, the bit-lines remain at $V_{bl,pre}$. If the FGT is on (i.e., there is no charge on the floating gate, which corresponds to a logical "1"), then the bit-line is pulled down. Otherwise, the FGT is off and the bit-line is not pulled down. Assuming $N_{bits_1}$ to be the number of bits corresponding to logical "1", the resulting energy dissipation is given by:

$$E_{bl-1,r} = 0.5 * C_{bl} * (0 - V_{bl,pre})^2 * N_{bits_1} \tag{3.9}$$

$$E_{sl,r} = E_{ssl,r} + E_{gsl,r} + E_{srcline,r} \tag{3.10}$$

where

$$E_{ssl,r} = E_{gsl,r} = 0.5 * C_{ssl} * (V_{read} - 0)^2$$
$$E_{srcline,r} = 0.5 * C_{srcline} * (V_{read} - 0)^2$$

The state of the cell is detected using the sense amplifier connected to each bit-line. The sense amplifier is a part of the page buffer and contains a latch unit [26]. It detects the voltage changes in the bit-line and compares it with a reference voltage. Since this is very similar to DRAM sense amplifier [49], we use CACTI's DRAM sense amplifier model to determine the dissipated energy $E_{senseamp,r}$.

Once the read operation is complete, the system transitions from the read state to the precharged state. This means that bit-lines corresponding to logical "1"are biased back to the precharge voltage $V_{bl,pre}$, while the select lines are biased back to ground from $V_{read}$ and the word-lines are precharged back to $V_{wl,pre}$. ( Note that bit-lines corresponding to logical "0" would not have discharged and hence it is not necessary to precharge them again). But the biasing for the transition from read operation to precharge state is equal but opposite to the biasing for the transition from the precharge state to read operation. Hence the energy to transition from the read to the precharge state ($E_{r-pre}$) is given by,

$$E_{r-pre} = E_{bl-1,r} + E_{selected-page,r} + E_{unselected-pages,r} + E_{sl,r} \quad (3.11)$$

We can calculate the energy dissipated by the decoders for the read operation, $E_{dec,r}$ ,using CACTI. We modify the CACTI decoder model so that for each read and program operation, two levels of decoding (block and page decode) are performed. Hence the total energy dissipated per read operation is

$$E_r = E_{selected-page,r} + E_{unselected-pages,r} + E_{bl-1,r} + E_{sl,r} + E_{r-pre} + E_{senseamp,r} + E_{dec,r} \quad (3.12)$$

### 3.4.5 The Program Operation

The program operation involves decoding the page to be programmed and transferring the data from the controller to the page buffers. Since the decoding for the program operation is the same as that of the read operation, the energy dissipated for decoding during the program operation is equal to $E_{dec,p} = E_{dec,r}$, where $E_{dec,r}$ was estimated using CACTI for read operation. After the data to be programmed is latched in the page buffers, the selected word-line is biased to $V_{pgm}$ while the un-selected word-lines are set to $V_{pass}$ to inhibit them from programming. Thus the energy dissipation for the selected page and the un-selected page is given by:

$$E_{selected-page,p} = 0.5 * C_{wl} * (V_{pgm} - V_{wl,pre})^2 \tag{3.13}$$

$$E_{unselected-page,p} = 0.5 * C_{wl} * (V_{pass} - V_{wl,pre})^2 * (N_{pages} - 1) \tag{3.14}$$

*FlashPower* adopts the self-boosted program inhibit model [45] to prevent cells corresponding to logical "1" from being programmed. This is achieved by boosting the channel voltage by biasing the bit-lines corresponding to logical "1" at $V_{bl,ip}$ and setting the SSL to $V_{dd}$. Assuming $N_{bits_1}$ to be the number of bits corresponding to logical "1", the energy is given by

$$E_{bl,ip} = 0.5 * (C_{bl} - C_{d,mc} * N_{pages}) * (V_{bl,p} - V_{bl,pre})^2 * N_{bits_1} \tag{3.15}$$

With a high voltage, $V_{pgm}$, on the word-line, the electric field across the channel and the floating gate gets high enough for FN tunneling to take effect. As electrons get tunneled from the channel to the floating gate, a tunneling current, $I_{FN}$, flows across the channel and the threshold voltage of the FGT increases by $\Delta V_{th}$to $V_{pref}$. To avoid wearing out of the tunnel oxide, the increase in threshold voltage is generally achieved by using multiple short pulses (referred to as sub-program operation) of $V_{pgm}$ applied to the control gate. The voltage pulse is applied for a short interval and after each interval, a verify-program operation is performed to check if $\Delta V_{th}$ increase is achieved. If the desired increase is not achieved, this process is repeated (for a maximum of $N_{loop}$ times) for

successful programming. Thus, the total program energy along the bit-line is calculated as,

$$E_{bl,p} = (0.5 * (C_{bl} - C_{d,mc} * N_{pages}) * (0 - V_{bl,pre})^2 + E_{tunnel,mc}) * (N_{bitlines-perblock} - N_{bits_1}) \quad (3.16)$$

where the term $E_{tunnel,mc}$ is the tunneling energy per cell. $E_{tunnel,mc}$ is calculated as

$$E_{tunnel,mc} = \Delta V_{th} * I_{FN} * t_{sub-program} \quad (3.17)$$

where $I_{FN}$ is the tunnel current and $t_{sub-program}$ is the duration of sub-program operation. $I_{FN}$ is calculated as $I_{FN} = J_{FN} * A_{fgt}$, where $J_{FN}$ is the tunnel current density calculated using [27] and $A_{fgt}$ is the area of the floating gate.

Then the energy dissipated in charging the select lines is given by:

$$E_{sl,p} = E_{ssl,p} + E_{gsl,p} + E_{srcline,p} \quad (3.18)$$

where

$$E_{ssl,p} = E_{gsl,p} = 0.5 * C_{ssl} * (V_{dd} - 0)^2$$
$$E_{srcline,p} = 0.5 * C_{srcline} * (V_{dd} - 0)^2$$

Hence, the total energy per sub-program operation is :

$$E_{subp} = E_{selected-page,p} + E_{unselected-page,p} + E_{bl,ip} + E_{bl,p} + E_{sl,p} \quad (3.19)$$

Since the entire process of sub-program and verify-program is repeated a maximum of $N_{loop}$ number of times, the maximum energy for programming is given by :

$$E_{pgm} = N_{loop} * (E_{subp} + E_{vp}) \quad (3.20)$$

where $E_{vp}$, the energy spent in verify-program. *FlashPower* currently models the verify-program as

a read operation. $N_{loop}$ is obtained from datasheets like [39] or can be fed as a input to the model.

Since a program operation concludes with a read operation, the transition from the Program to Precharge is same as the transition from a read to precharge.

$$E_{p-pre} = E_{r-pre} \tag{3.21}$$

where $E_{r-pre}$ is given by equation (3.11).

Hence the total energy dissipated in the program operation is equal to:

$$E_p = E_{pgm} + E_{p-pre} + E_{dec,p} \tag{3.22}$$

### 3.4.6 The Erase Operation

Since erasure happens at a block granularity, the controller sends the address of the block to be erased. Only the block decoder is used and the energy dissipated for block decoding ($E_{dec,e}$) is calculated using CACTI. To aid block-level erasing, the blocks are physically laid out such that all pages in a single block share a common P-well. Moreover, multiple blocks share the same P-well [45] and therefore it is necessary to prevent other blocks sharing the same P-well from being erased. *FlashPower* assumes the self-boosted erase inhibit model [45] to inhibit other blocks sharing the same P-well from getting erased.

For the erase operation, the control gates of all the word-lines in the selected block are grounded. The P-well for the selected block is biased to erase voltage $V_{era}$. The SSL and the GSL are biased to $V_{era} * \beta$ where $\beta$ is the capacitive coupling ratio of cells between control gate and the P-well. Typical value of $\beta$ is 0.8 [7]. The SL is biased to $V_{bl,e}$. Here $V_{bl,e}$ is equal to $V_{era} - V_{bi}$ where $V_{bi}$ is the built-in potential between the bitline and the P-well of the cell array [38]. Adding up the charging of the SSL, GSL and SL, we get the energy dissipated in the select lines to be:

$$E_{sl,e} = E_{ssl,e} + E_{gsl,e} + E_{srcline,e} \tag{3.23}$$

where $E_{ssl,e}, E_{gsl,e}$ and $E_{srcline,e}$ are calculated using the SSL, GSL and SL capacitance and the

bias condition explained above.

The bit-lines are biased to $V_{bl,e}$ which dissipates energy that is modeled as,

$$E_{bl,e} = 0.5 * C_{bl} * (V_{bl,e} - V_{bl,pre})^2 * N_{bitlines-perblock} \tag{3.24}$$

With the P-well biased to $V_{era}$, cells that were 0-programmed earlier undergo FN tunneling. Electrons tunnel off the floating gate onto the substrate and the threshold voltage of the cell is reduced to the level of 1-programmed cells. It is assumed that 1-programmed cells in the same block do not undergo tunneling because their threshold voltage is already low and tunneling would further reduce this.

To effect FN tunneling, the depletion layer capacitance between the P-well and the N-well should be charged. This capacitance of the junction between the P-well and the N-well, which form a P-N junction, is a function of the applied voltage $V_{era}$, the area of the p-well $A_{well}$. The dynamic power dissipated to charge this capacitance as the voltage across the junction raises from $0V$ to $V_{era}$ is determined as:

$$E_{junction,e} = (\frac{C_{j0}}{(1 - V_{era}/\phi_0)^m} * A_{fgt}) * V_{era}^2 \tag{3.25}$$

where $C_{j0}$ is the capacitance at zero-bias condition and $\phi_0$ is the built-in potential of the P-N junction. $m$, the grading coefficient is assumed to be 0.5.

Once this P-N junction capacitance is fully charged, tunneling happens in all 0-programmed cells. The tunneling energy is calculated assuming that the duration of the $V_{era}$ pulse is same as $t_{sub-program}$. Hence the total energy due to tunneling of all 0-programmed cells in the array is calculated as,

$$E_{tunnelerase,mc} = E_{tunnel,mc} * (N_{bitlines-perblock} - N_{bits_1}) \tag{3.26}$$

where $E_{tunnel,mc}$ is given by equation (3.17).

After a block is erased, a verify erase operation is performed to ensure that all FGTs in the block are erased [45]. The verify erase operation is a single read operation which consumes energy

equivalent to a read operation. This is modeled as $E_{block,ve} = E_r - E_{dec,r}$ , where $E_r$ is given by equation (3.12). Since an erase operation ends with a read operation, the transition from the erase to precharge is same as the transition from read to precharge.

$$E_{e-pre} = E_{r-pre} \tag{3.27}$$

where $E_{r-pre}$ is given by equation (3.11).

Hence the total energy dissipated in erase operation $E_{erase}$ is equal to:

$$E_{erase} = E_{dec,e} + E_{bl,e} + E_{tunnelerase,mc} + E_{junction,e} + E_{sl,e} + E_{e-pre} + E_{block,ve} \tag{3.28}$$

### 3.4.7 I/O pins

Since a read/program operation involves transfer of data to/from the controller to the chip, the I/O pins in the chip switch continuously during the data transfer which dissipates energy. We estimate I/O pin energy as,

$$E_{iopin} = C_{pin} * V_{dd}^2 * N_{bitlines-perblock} \tag{3.29}$$

where $C_{pin}$ is the capacitance of a pin that needs to be driven, $V_{dd}$ is the applied voltage to the pin.

### 3.4.8 Multi-plane operation

The power model described thus far corresponds to single-plane operations. However modern NAND flash chips allow multiple planes to operate in parallel. These are referred to as multi-plane operations. Since *FlashPower* models single-plane operations, energy consumption for multi-plane operations can be determined by multiplying the results of single-plane operations with the number of planes operating in parallel.

### 3.4.9 Charge pumps

We also model the energy dissipated by charge-pumps during the program and erase operations. The energy consumed by charge-pumps is provided in [19], where the authors specify that conventional

charge-pumps operating at 1.8V consume 0.25$\mu$J while charge-pumps operating at 3.3V consumed 0.15$\mu$J. We use these constant values in *FlashPower*, but since *FlashPower* is parameterized, a detailed charge-pump model can be easily incorporated in lieu of the current one.

### 3.4.10 Limitations of FlashPower

Even though *FlashPower* provides a detailed power model for SLC NAND flash memory, it has two limitations. *FlashPower* currently does not model the energy consumption of MLC NAND flash memories and NOR flash memories. *FlashPower* can be extended to model MLC-NAND flash by increasing the number of verify-program cycles for the program operation and modeling sense-amplifiers that can detect multiple voltage levels for read operation. We plan to extend *FlashPower* to include power models for MLC chips and NOR flash in future work.

# Chapter 4

## Validation of *FlashPower*

Validating *FlashPower* is challenging since there is very little publicly available information on the power consumption of flash chips and manufacturer datasheets are not sufficient for validation purposes. Interestingly, researchers at the University of California, San Diego recently published a paper on an empirical study they undertook to understand the characteristics of flash chips [14]. This study includes power measurements. We validate *FlashPower* by comparing the power estimates provided by our model to the measured values reported in [14]. This paper reports the energy dissipation of a single plane and includes the data transfer between the page buffer and the I/O pins of the flash chip. The data provided in that paper regarding the microarchitecture of flash chips is given in Table 4.1.

The chips are from two different manufacturers which are denoted as [14]: "A" and "B". In addition to the data given in the table, we require additional details about the microarchitecture of those chips and the flash device technology used in them to calculate the energy dissipation in

| Chip Name | Capacity (Gb) | Pagesize + Spare area (B) | Pages/ Block | Blocks/ Plane | Planes/ Die | Dies |
|---|---|---|---|---|---|---|
| A2 | 2 | 2048+64 | 64 | 1024 | 2 | 1 |
| A4 | 4 | 2048+64 | 64 | 4096 | 1 | 1 |
| A8 | 8 | 2048+64 | 64 | 4096 | 2 | 1 |
| B2 | 2 | 2048+64 | 64 | 2048 | 1 | 1 |
| B4 | 4 | 2048+64 | 64 | 2048 | 2 | 1 |

Table 4.1: Microarchitectural parameters of the validation chips.

*FlashPower*. These unknown parameters, which tend to be manufacturer specific, include:

- Device feature sizes
- Applied bias voltages for the read, program, and erase operations
- Capacitance of the chip I/O pins, which can vary from 5 pF-40 pF across chips and manufacturers
- The number of partial program/erase cycles, which can be as high as 4-8 cycles across chips and manufacturers
- Manufacturer-specific device technology and circuit optimizations

In our validation study, we assume the feature size to be 80nm. In addition to the feature size, *FlashPower* also requires other device technology information such as the doping-levels of the P- and N-wells, estimates for which we obtained from industry [38]. The values for the various voltage bias parameters were obtained from [20, 7]. The value for the capacitance of the I/O pins and the latencies for the various flash operations were obtained from [39].

Another key factor that can influence the energy dissipation is the mix of 0's and 1's that are read from, written to, or stored in the flash cells (in the case of erase). The read energies are only slightly affected by the mix of 0's and 1's because the read energy is determined by the number of bit-lines that are discharged, which depends on the number of cells that hold a "1", but is more strongly affected by the energy required to transfer the data from the page buffer to the I/O pins, which is relatively unaffected by the value of the bits. On the other hand, the program energies are heavily impacted by the bit values because writing a "0" to a flash cell requires driving its corresponding bit-line and therefore the bit-line energy scales up with the number of 0's that need to be written to a page. Since we do not know the mix of 0's and 1's in the workloads used for the power measurements, we present the energy dissipation data for the case where half the bits are 0's and the other half are 1's as well as cases where the bit distribution causes the least or the most amount of energy to be dissipated for the read, program, and erase operations.

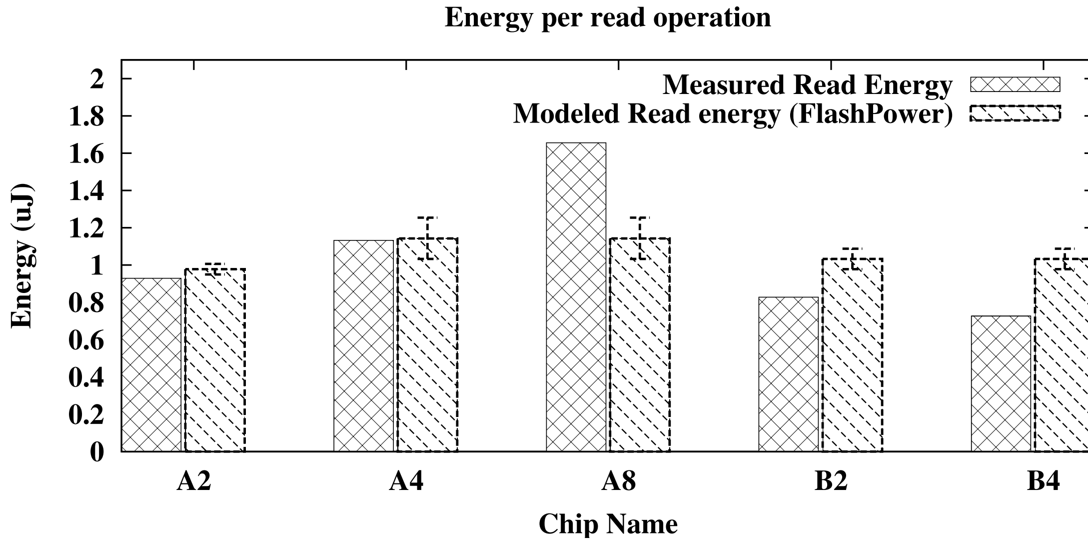



Figure 4.1: Validation Results for Read Operation

The validation results are given in Figures 4.1, 4.2 and 4.3. For each of the three flash operations, we present the power measurement data from [14] as well as estimates provided by *FlashPower*. The error-bars in Figures 4.1,4.2 and 4.3 depict the energy variation based on the mix of 0's and 1's read from/written to the cells of a page during reads and programs, or stored within the cells of a block during erase. Since the distribution of 0's and 1's and the number of partial program/erase cycles both impact the energy dissipation and their values for the actual power measurements are not reported in their paper [14], we vary both these parameters in our validation experiments to ascertain whether the measured data is close to the estimated energy within this range of uncertainty. For the read operation shown in Figure 4.1, the first bar in each pair corresponds to the measured value whereas the second bar corresponds to the energy estimate provided by *FlashPower* assuming that an equal number of 0's and 1's are read from the page. The error-bar is to account for the range of possibilities with regard to the bit distribution; the lower edge of the bar corresponds to a page with all 0's and the upper edge to all 1's. For the write operation, shown in Figure 4.2, the energy dissipation depends on the number of partial program cycles and also on the mix of 0's and 1's written to the page. We consider two values for the partial program cycles (2 and 4 cycles), which correspond to the second and third bars in the triplet of bars for each chip. The error-bars for each

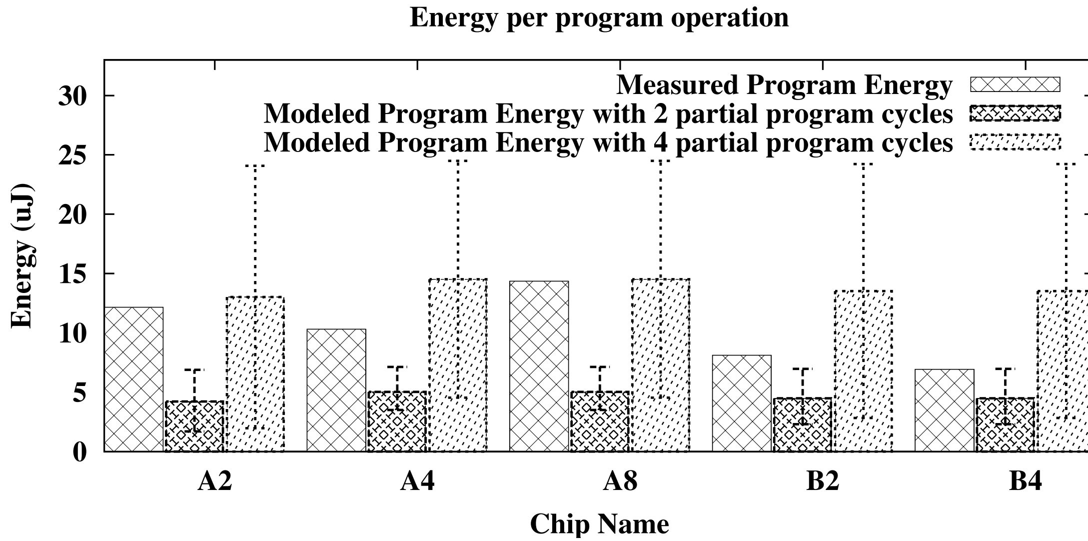



Figure 4.2: Validation Results for Program Operation

of the estimates show the range of energy dissipation values from a pattern of all 1's (least energy dissipation) to all 0's (highest energy dissipation). For the erase operation (Figure 4.3), the energy dissipation depends on the number of cells that need to undergo tunneling, which in turn depends on the bits they contain when the erase operation is initiated. The second bar in each pair corresponds to the energy dissipation for the case where the cells contain an equal number of 0's and 1's and the error-bar shows the variation in energy for cells that contain all 1's to those that contain all 0's. *Note that the error-bars do not indicate a limitation of the model.* Instead, these bars represent the range of values that a workload may generate or access, which will be input to *FlashPower* by an architecture simulator.

Overall, we observe that *FlashPower* is able to estimate the energy characteristics of the three chips from manufacturer "A" with reasonable accuracy for all three operations within the range of uncertainty imposed by the bit distributions and the number of partial program and erase cycles. The only exception is chip A8, where the estimated read energy is approximately 0.35 nJ lower than the measured value. We believe that this difference is due the I/O pin capacitance of this chip. As mentioned previously, the pin capacitance tends to vary from 5-40 pF between different chips, even for the same manufacturer. We find that even a small change in the capacitance that

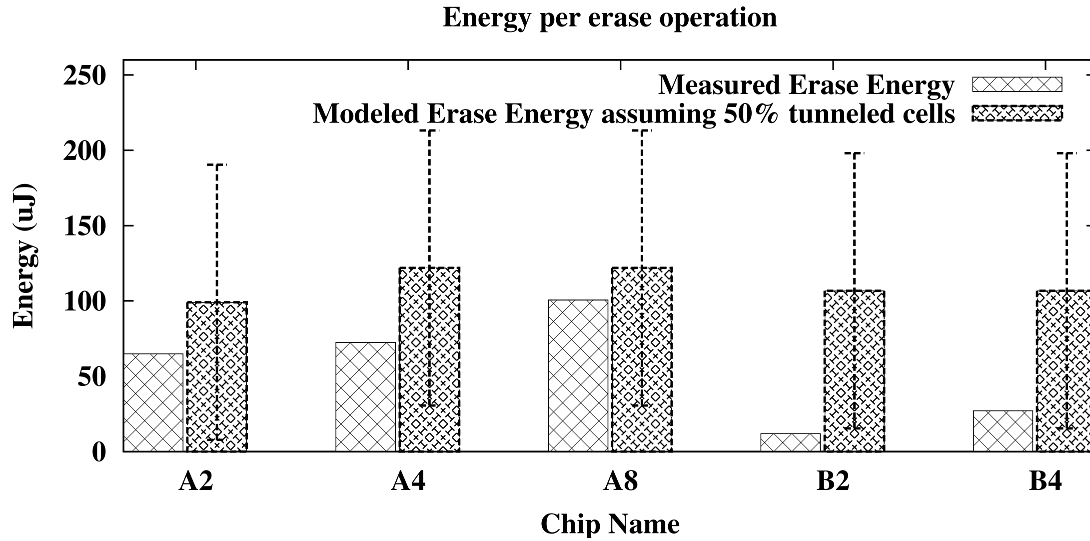



Figure 4.3: Validation Results for Erase Operation

we assume can have a significant impact on the read energy. For example, find that changing the pin capacitance from 5 pF (our default assumption for all the chips) to just 7.5 pF shifts the bar upwards and brings the measured energy dissipation value for A8 within the range of possible values estimated by *FlashPower*. However, our model is unable to accurately estimate the energy characteristics of chips from manufacturer "B", especially for reads and erase. As mentioned earlier, there are several manufacturer-specific implementation details that can have a significant impact on the energy estimation and it is possible that the values that we chose as inputs to the model, a few of which were drawn from a specific datasheet [39], do not match the characteristics of these chips.

The larger range of read energy dissipation values for A4 and A8 compared to A2 is due to the fact that they are higher-capacity devices and therefore the cells that store 1's have to discharge via longer bit-lines. For the program operation, the energy dissipation increases when the programming latency is higher due to an increase in the tunneling energy. The larger range of energy dissipation values for the longer latency program operation is due to the fact that a single page in these devices consists of $2^{14}$ cells and the energy dissipation depends on how many of those cells tunnel based on the bits they store. Similarly, we can observe that the erase operation dissipates the most energy and also shows a large variation in the dissipation since a block consists of $2^{20}$ cells. This large

variation in the energy dissipation based on the actual values of the bits suggests opportunities for applying architecture level power optimization techniques such as invert coding [44]. *FlashPower* can be used to evaluate such techniques.

# Chapter 5

# *FENCE*: An Endurance Model for NAND Flash

Having explained the theory behind the power model and having validated the power model with real chip measurements, we now turn our attention to the endurance model of NAND flash which forms the second part of the thesis. As mentioned in Chapter 2, data in a memory cell is sensed by detecting a shift in threshold voltage of an FGT. This shift occurs due to the addition/removal of charges to/from the floating gate of the FGT through Fowler-Nordheim (FN) tunneling [27]. These operations are referred to as program and erase operations. These operations are *stress events* that have a detrimental impact on the reliability of flash memory as they affect both the *retention* and the *endurance*.

The typical data retention time for flash memory is 10-20 years [20]. However, as a flash memory cell is repeatedly programmed and erased, the tunnel oxide layer becomes weak which leads to an increase in the Stress Induced Leakage Current (SILC) of the memory cell, thus affecting data retention. While P/E cycles clearly play a role, it has been shown that a major factor that affects retention is high temperature [28].

Endurance is a measure of the number of P/E cycles that a flash memory cell can tolerate while preserving the integrity of the stored data, and is a function of the charge trapping characteristics of the tunnel oxide [51, 30]. Every stress event increases the likelihood of charges getting trapped in the tunnel oxide, which can lead to an undesirable increase in the threshold voltage of the memory cell. Since FGTs have a programmable threshold voltage, the threshold voltage of the flash memory cell at a given time directly corresponds to how the data stored in the cell is interpreted. Therefore,

if a sufficiently high number of charges get trapped in the oxide, it will no longer be possible to reliably read the cell.

Although a memory cell that undergoes a large number of stress events will have more charges trapped in its tunnel oxide, several transistor-level studies of NAND-flash memory have shown that it is possible to *detrap (i.e., remove)* some of the charges from the tunnel oxide under certain conditions [51, 30, 47]. The detrapping of charges from the oxide has a self-healing effect on the memory cell improving its endurance. Beneficial conditions for detrapping include higher external temperatures and quiescent periods between successive stress events. Furthermore, the measurement studies indicate that introducing a quiescent period to allow detrapping can be applied at temperatures as low as $25°C$, which is the typical external ambient temperature of a disk [17]. Since the quiescent periods help improve endurance, we refer to them as *recovery periods*.

## 5.1 FENCE: An Architecture-Level Model for Estimating Endurance

Since studying the impact of charge trapping and detrapping in FGTs is a relatively unexplored area at the architecture level, we have developed an analytical reliability model that is suitable for use in an architectural simulation of a SSD. In order to analyze how stress events and recovery periods impact the endurance of NAND flash memory under various usage scenarios, we constructed *Flash EnduraNCE (FENCE)*, an analytical model that captures how these two parameters affect the threshold voltage of memory cells. *FENCE* was constructed by synthesizing information from device physics publications for sub-90nm NAND flash memory cells [51, 29, 30, 52]. These papers provide information about how these parameters are related and also provide memory cell level measurements of stresses and recovery for a range of values.

*FENCE* consists of two parts - the first part models stresses while the other part models recovery. The first part of the model gives the relationship between the increase in threshold voltage due to charge trapping ($\Delta V_{th,s}$) and the number of stress events on the tunnel oxide. The second part gives the relationship between the threshold voltage shift due to recovery ($\Delta V_{th,r}$), the amount of trapped charges in the tunnel oxide due to stress calculated in the first part ($\Delta V_{th,s}$), and the recovery period

(t). Using these two parts, we calculate the effective increase in threshold voltage of a memory cell due to trapped charges ($\delta V_{th}$) after a stress event and a subsequent recovery period. We now explain these two components of the model in more detail.

### 5.1.1 The Stress Model

The threshold voltage of a memory cell increases due to charge trapping with the number of stress events (program or erase cycles) [52]. There are two types of traps that form in the tunnel oxide - interface traps and bulk traps - both of which contribute to the increase in the threshold voltage. It has been shown that both these traps have a power-law relation to the number of cycles on the memory cell [52] as:

$$\Delta N_{it} = A * cycle^{0.62}$$

$$\Delta N_{ot} = B * cycle^{0.30}$$

where $A$ and $B$ are constants, $cycle$ is the number of program or erase cycles on the cell, and the terms $\Delta N_{it}$ and $\Delta N_{ot}$ are the interface and bulk trap densities respectively. In addition to providing this power-law relationship, [52] also provides empirical data on how $\Delta N_{it}$ and $\Delta N_{ot}$ vary with $cycle$. We calculated the values of constants $A$ and $B$ to be 0.08 and 5 respectively for the sub-90nm process technology from this empirical data. Similar device characterization data can be used for other process technologies in our model to estimate endurance

The total threshold voltage increase due to trapping is divided into interface trap voltage shift ($\Delta V_{it}$) and bulk trap voltage shift ($\Delta V_{ot}$). Park et al. [35] give the relationship between $\Delta V_{it}$ and $\Delta N_{it}$ and between $\Delta V_{ot}$ and $\Delta N_{ot}$ to be:

$$\Delta V_{it} = \frac{\Delta N_{it} * q}{C_{ox}} \tag{5.1}$$

$$\Delta V_{ot} = \frac{\Delta N_{ot} * q}{C_{ox}} \tag{5.2}$$

where $q$ is electron charge ($1.6 \times 10^{-19}$ Coulombs) and $C_{ox}$ is the capacitance of the tunnel oxide.

The value of $C_{ox}$ depends on the feature size of the NAND flash cell.

Hence the increase in threshold voltage of the memory cell due to trapped charges, $\Delta V_{th,s}$, is given by

$$\Delta V_{th,s} = \Delta V_{it} + \Delta V_{ot} \tag{5.3}$$

where $\Delta V_{it}$ and $\Delta V_{ot}$ are given by equations (5.1) and (5.2).

### 5.1.2 The Recovery model

According to Yamada et al. [51], the threshold voltage shift due to detrapping depends on the recovery period and the amount of charge trapped in the oxide. This relationship is given by the equation

$$\Delta V_{th,r} = c_{vt}.ln(t) \tag{5.4}$$

where $t$ is the recovery period between successive stress events to the *same cell* (in seconds) and $c_{vt}$ depends on the amount of trapped charge ($Q$) present in the oxide. The value of the recovery period, $t$, is assumed to be finite and greater than 1 second. We conservatively assume that no charge detrapping occurs for recovery periods less than one second. Yamada et al. [51] also show that $c_{vt}$ has a logarithmic dependence on $Q$. Since $Q$ is directly proportional to the stress voltage, $\Delta V_{th,s}$, $c_{vt}$ also has a logarithmic dependence on $\Delta V_{th,s}$. Therefore, we get

$$c_{vt} \propto ln(\Delta V_{th,s}) \tag{5.5}$$

Equation (5.5) can be rewritten as

$$c_{vt} = K * ln(\Delta V_{th,s}) \tag{5.6}$$

where $K$ is a constant that denotes the efficiency of the recovery process. [51] also provide plots of how $c_{vt}$ varies with $\Delta V_{th,s}$.

Combining equations (5.4) and (5.6), the change in the threshold voltage shift due to recovery is given by:

$$\Delta V_{th,r} = K * ln(\Delta V_{th,s}).ln(t) \tag{5.7}$$

where $\Delta V_{th,s}$ is given by equation (5.3). We assume the value of K to be 60% based on discussions with industry [31]

The effective increase in the threshold voltage due to trapped charges after stress and recovery of the tunnel oxide, $\delta V_{th}$, is given by

$$\delta V_{th} = \Delta V_{th,s} - \Delta V_{th,r} \tag{5.8}$$

Equations (5.3) and (5.7) can be used to estimate the endurance of a NAND flash memory cell based on the number of stress events (P/E cycles) and the recovery periods that the cell experiences. The stress events and recovery periods can be tracked over the course of an architectural simulation using a simulator such as Disksim [13].

**Granularity of Estimating Endurance:** While the model captures the impact of stress and recovery on a single memory cell, the program and erase operations in NAND flash occur for a group of cells, such as within a page (for program) or within a block (for erase). Therefore, we track stress events due to program and erase operations at the granularity of a page and block respectively. (For erase operations, we assume every page within a block undergo a stress event).

**SLC vs. MLC Cells:** In a flash memory array, a memory cell can store a single bit of data (SLC) or multiple bits of data (MLC). However, the phenomenon of charge trapping and detrapping applies to flash memory cells irrespective of whether they are used in the SLC or MLC mode. In either case, one can use the methodology given above to derive models from their physical device-level characterizations to estimate their endurance.

### 5.1.3 Limitations of FENCE

Currently, FENCE has two limitations:

- FENCE does not capture the impact of temperature on stress and recovery. The constants in our model are estimated based on published datapoints for $25\,^{\circ}C$, which is approximately the external ambient temperature of a disk drive in a server with a well-designed cooling system [17, 41]. There is evidence that higher temperatures can accelerate the detrapping process [29, 30], but can exacerbate other aspects of SSD reliability such as flash cell retention.
- If the memory cell is used in the MLC mode, programming each n-bit value requires a different amount of time [7] and hence the duration of the stress event would be different. Currently, the model does not account for these variations and estimates the impact of stress and recovery in a way that is agnostic to the actual bits stored in the memory cells.

Despite these limitations, FENCE captures the primary effects of charge trapping and detrapping on the endurance of NAND flash and is suitable for use in architecture-level evaluations. The next section describes how we use FENCE to determine the impact of stress and recovery.

## 5.2 Impact of Charge Detrapping on SSD Endurance

Having derived a model for the threshold voltage shift due to stress and recovery, we now analyze the impact of charge detrapping on flash memory cells over different timescales. The goal of this analysis is to ascertain the extent to which charge detrapping can improve the reliability of flash memory cells by delaying endurance related failure and understand how the duration of the quiescent period affects the extent of the recovery.

Before we begin the analysis, we first need to precisely define what "failure" means with respect to endurance. The data stored in a flash memory cell is identified by a specific voltage level. An n-bit MLC has $2^n$ *distinct* voltage levels, each of which corresponds to an n-bit value (an SLC flash cell is merely the case where n=1, which corresponds to two voltage levels - one for a digital "0" and the other for a "1"). Let $\Delta V_{th,spread}$ be the threshold voltage range for a single voltage level

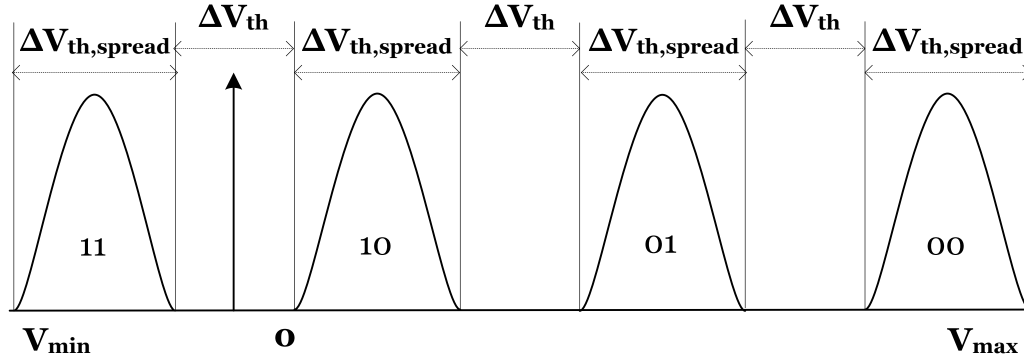



Figure 5.1: Threshold voltage distribution for a 2-bit MLC

in a memory cell and $\Delta V_{th}$ be the difference in voltage between adjacent levels. Then, the entire operating voltage range of a memory cell varies from $V_{min}$ to $V_{max}$, where $V_{max} = V_{min} + (\Delta V_{th,spread} * 2^n) + \Delta V_{th} * (2^n - 1)$. This is illustrated in Figure 5.1 for $n = 2$ (2-bit MLC).

When the charges trapped in the oxide result in a threshold voltage increase of $\Delta V_{th}$ or higher, it will no longer be possible to clearly distinguish between different voltage levels. As a consequence, it will not be possible to reliably read from or write to the memory cell. We define this situation where the increase in threshold voltage due to trapped charges, $\delta V_{th}$, is greater than or equal to $\Delta V_{th}$ as a *failure*. We define the *endurance limit* as the total number of P/E cycles before this condition occurs. Once the endurance limit is reached, a page is considered to have failed and is no longer usable.

The higher the $\Delta V_{th}$, the larger the number of P/E cycles required for $\delta V_{th}$ to reach this value, also, the longer the recovery period between successive P/E cycles, the higher the detrapping and therefore a larger number of P/E cycles will be allowed before $\delta V_{th}$ reaches $\Delta V_{th}$. It should be noted that while it appears that choosing a large $\Delta V_{th}$ can provide high endurance, a high threshold voltage directly translates to a higher write latency [7], which can significantly degrade performance and actually increase the duration of stress.

Manufacturer datasheets specify an endurance limit of 10K and 100K P/E cycles for MLC and SLC chips respectively. However, these values specify the *minimum* number of P/E cycles that the chip is expected to tolerate before failure, tested under high stress conditions where the flash

cells are continuously erased and rewritten with little or no recovery time between successive stress events [47]. There is anecdotal evidence in recently published papers on measurements of NAND flash chips [14, 11], that, in the common case, when there are recovery periods between the stress events, the endurance of flash is higher than the values specified in datasheets.

In Figure 5.2, we plot the change in $\delta V_{th}$ with the number of P/E cycles, over a number of timescales for the recovery period, for the 80nm process technology [20] for which published memory cell characterization data is available [51, 30, 52]. We consider the case where there is no recovery between successive stress events, which is how the datasheet values are computed, and also cases where the recovery time is varied from 10 seconds to over 2 days.

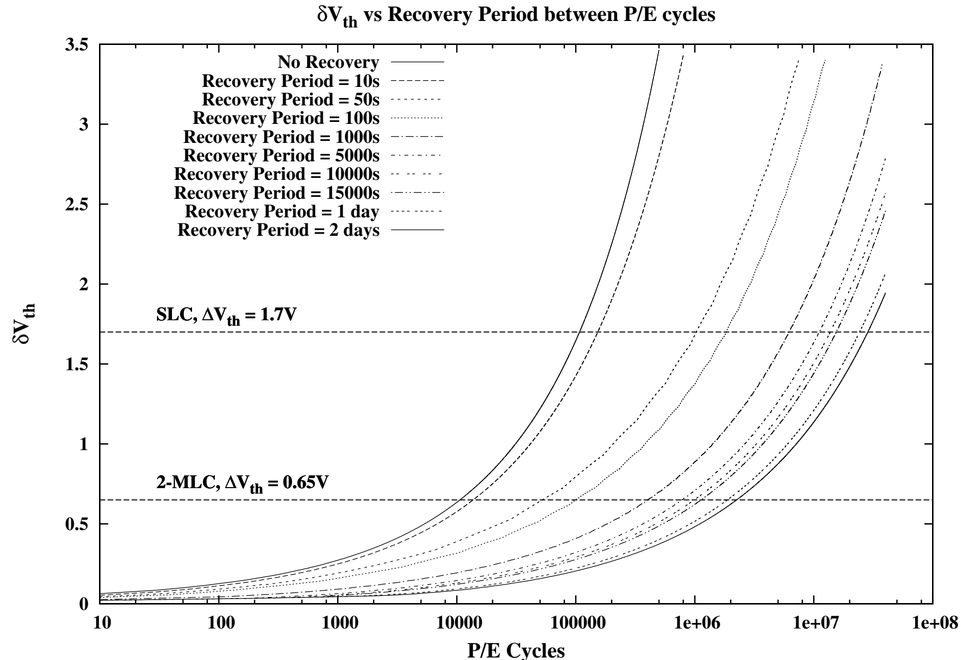



Figure 5.2: Increase in $\delta V_{th}$ with P/E cycles for different recovery periods.

To illustrate how these curves translate to endurance, we plot the $\Delta V_{th}$ for SLC and 2-bit MLC flash. These values are shown as horizontal lines in the graph and are obtained from threshold voltage distributions of prototype NAND flash memory chips published in the literature. 2-bit MLC devices have $\Delta V_{th}$ values that are approximately equal to $\Delta V_{th,spread}$ and have been shown to vary from 0.6V to 0.7V [10]. We assume $\Delta V_{th}$ to be equal to $\Delta V_{th,spread}$ for SLC devices as

| **Recovery Period** | SLC, $\Delta V_{th} = 1.7V$ | | 2-bit MLC, $\Delta V_{th} = 0.65V$ | |
|---|---|---|---|---|
| | **P/E Cycles** | **Endurance Increase** | **P/E Cycles** | **Endurance Increase** |
| No recovery | 107535 | 1x | 10652 | 1x |
| 10 seconds | 153186 | 1.4x | 13749 | 1.3x |
| 50 seconds | 1028724 | 9.6x | 52444 | 4.9x |
| 100 seconds | 1837530 | 17.7x | 99913 | 9.3x |
| 1000 seconds | 6214983 | 57.8x | 403082 | 37.8x |
| 5000 seconds | 11093823 | 103x | 780723 | 73.3x |
| 10000 seconds | 13753999 | 127x | 990014 | 92.9x |
| 15000 seconds | 15497892 | 144x | 1129379 | 106x |
| 1 day | 24274492 | 225x | 1879352 | 176x |
| 2 days | 28487539 | 264x | 2247910 | 211x |

Table 5.1: Endurance limits with charge detrapping.

well. The $\Delta V_{th,spread}$ of SLC has been reported to vary from 1.4V to 2.0V [32, 25]. Based on this data, we assume the $\Delta V_{th}$ of SLC to be 1.7V and 2-bit MLC to be 0.65V. The portions of the curves below the horizontal lines correspond to failure-free operation of the cell. The number of P/E cycles attainable for each recovery period and the improvement in endurance over the case where there is no detrapping between successive stresses is given in Table 5.1 (for clarity, in the figure we omit a few of the datapoints given in the table).

We can see that when there are no recovery periods, the P/E cycles for the SLC and MLC datapoints approximately match the values given in datasheets (100K and 10K P/E cycles respectively), which concurs with the expected behavior. We can also see that a recovery period between successive P/E cycles can significantly boost endurance, which concurs with recent flash chip measurement studies [14, 11]. Even a recovery period of approximately a minute between stress events can provide a large improvement in endurance. Note that although the I/O request inter-arrival times to an SSD tend to be much shorter in an enterprise system and a significant fraction of the requests can be writes [33], the time between successive stress events to a *specific physical flash page* on the SSD is much longer due to the fact that NAND flash does not support in-place writes, due to the decisions made by the wear-leveling and cleaning policies of the FTL, and the logical block addresses in the I/O request stream arriving at the SSD. Further increase in the duration of the recovery periods provides increased endurance, and a recovery period of a few hours provides

two orders of magnitude improvement. However, as the recovery periods increase beyond a day, we start getting diminishing endurance benefits.

These results clearly show that charge detrapping can significantly boost endurance, a phenomenon that prior architecture and system-level studies did not consider.

**The Role of Error-Correcting Codes (ECC):** Modern SSD controllers use error correcting codes to detect and correct bit errors that occur due to various reliability phenomena, such as transient errors due to read and program disturbs, as well as endurance. While ECC is a reactive technique to handle errors in memory cells once they have occurred, charge detrapping proactively improves endurance of flash memory cells, thereby reducing the number of errors that occur. ECC is still necessary to handle other reliability issues such as disturb events and therefore detrapping and ECC are complementary in how they boost flash memory reliability.

# Chapter 6

## Analysis of SSD-Level Endurance

Having examined the impact of detrapping on endurance at a transistor level, we next use this model to determine the endurance of SSDs when stressed under real realistic data center usage scenarios. To simulate a SSD based storage system, we use Disksim [13], a widely used trace driven simulator for studying storage systems. We use the Disksim SSD extension [2] that facilitates studying a variety of SSD designs. We simulate a 32GB SSD composed of 8 4GB SLC NAND flash chips, similar to enterprise-class SSDs currently available in the market today [18]. The configuration of our SSD is summarized in Table 6.1.

Our workloads consist of block-level I/O traces collected from various production systems within Microsoft data centers that are publicly available via the IOTTA trace repository [43]. We present the results for four representative workloads: LM, EXCH, RAD, and MSNFS. LM

| Parameter | Value |
|---|---|
| Page Size | 4KB |
| Pages per block | 64 |
| Blocks per plane | 2048 |
| Planes per Die | 4 |
| Dies per package | 2 |
| Number of packages | 8 |
| Over-provisioning | 10% |
| Page Read | 25 $\mu s$ |
| Page Program | 200 $\mu s$ |
| Block Erase | 1.5ms |
| Serial Data Transfer - ONFI 2.0 | 25 $\mu s$ |

Table 6.1: SSD Configuration Used for Evaluation.

is a trace from a back-end server that hosts Live Maps™. The EXCH trace is obtained from an Exchange™ mail server. RAD is a trace of I/O requests to a RADIUS™ corporate authentication server. MSNFS is a trace from a MSN™ storage metadata and file server. A detailed description of these traces is given in [36]. Each workload consists of several sub-traces, each of which correspond to the I/O activity during a specific interval of time (e.g., an hour) on a typical day, and the collection of these sub-traces span at least one full day. We use all the sub-traces of each workload in the simulation to characterize the variations in the I/O behavior and their impact on the SSD over a one-day period.

**Endurance Metric:** We report endurance in terms of the number of P/E cycles. We use P/E cycles since SSD vendors typically use this metric as an indicator of endurance. Since different blocks may undergo a different number of P/E cycles over the course of time based on the workload and FTL behavior, we report the average number of P/E cycles and the minimum and maximum values observed across all the blocks. We report endurance at the end of 5 years of activity. This 5-year service life allows us to examine the impact of P/E cycles on endurance well before retention related reliability problems arise (the retention period of flash is 10-20 years [20]).

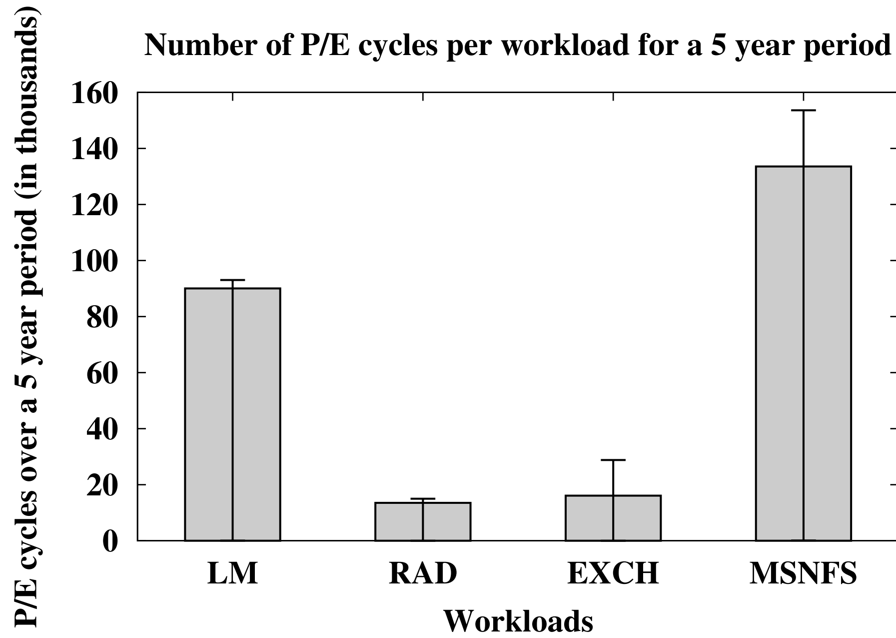
**Estimating Endurance Over the Service Life:** Since the service life spans multiple years whereas the traces record only a single day of activity, we need a way of estimating the activity on the SSD over this long time period. Since each trace represents the I/O activity over the course of a typical day, one approach could be to repeatedly replay the trace in Disksim and simulate 5 years worth of activity. However, this approach would require excessively long simulation times. We instead use a statistical approach to estimate the I/O activity on the SSD.

In order to estimate endurance, we need to capture two aspects of stress behavior: (1) the distribution of stress events across various pages and blocks in the SSD (spatial behavior), and (2) the distribution of the recovery periods to individual pages and blocks (temporal behavior). To determine these distributions, we collect an output trace over the course of a Disksim simulation that records when a particular page or block within a certain flash chip is programmed or erased. (We do not record reads to a page since read operations have a negligible impact on endurance). We collect one such output trace for each sub-trace that we simulate for each workload. Collecting an

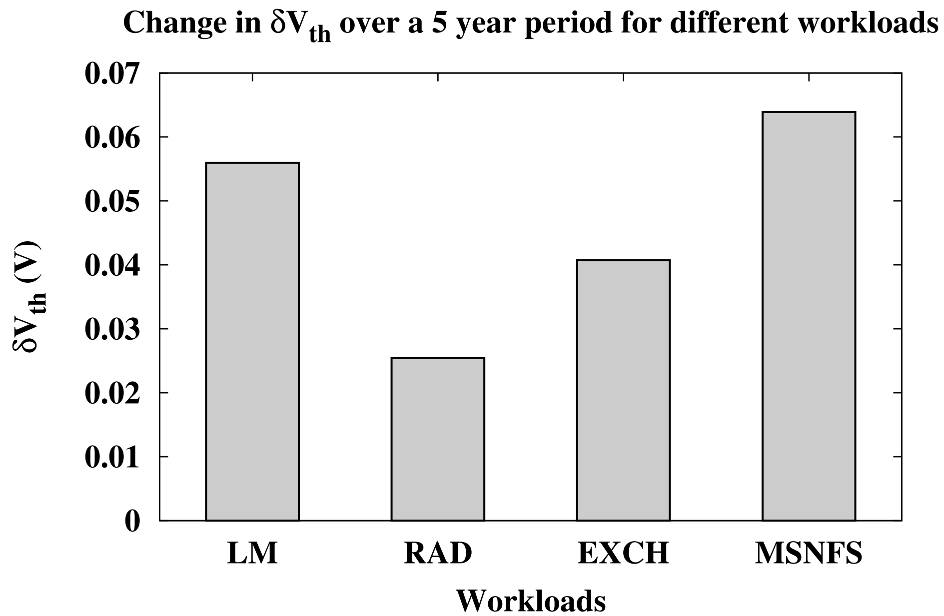
output trace per workload sub-trace allows us to capture any phase behavior within a workload and its consequent impact on stress patterns within the SSD. From this output trace, we characterize the spatial behavior of the workload by creating a histogram of the stresses to the different flash chips in the SSD to determine the frequency at which pages/blocks within a particular chip are stressed. Since wear-leveling operations are performed within each flash chip in an SSD [2] and the objective of wear-leveling is to spread out the stress events uniformly across all the pages and blocks within the chip, we use an uniform distribution to model the pattern of stresses within a chip. We characterize the temporal behavior of the workload by creating a histogram of the recovery periods of all the pages within the SSD. Using these statistical distributions of the spatial and temporal characteristics of a workload's stress behavior on the SSD, we extrapolate the stress behavior over the service life of 5 years.

## 6.1 Results

The number of P/E cycles over the 5-year service life of the SSD for each workload is given in Figure 6.1(a). Each bar gives the average number of P/E cycles across all the SSD blocks while the error-bar shows the smallest and largest number of P/E cycles observed across the blocks. We can see that there is significant variation in terms of the number of P/E cycles that blocks experience across the different workloads over the SSD service life. The RAD and EXCH workloads impose fewer stresses on the blocks whereas LM and MSNFS impose far greater number of stresses. The change in $\delta V_{th}$ for the block that experienced the largest number of P/E cycles for each workload is given in Figure 6.1(b). We observe that across all the workloads, the increase in $\delta V_{th}$ over the 5-year period is well below the SLC $\Delta V_{th}$ of 1.7V. This is true even for LM and MSNFS whose P/E cycles are close to or greater than the datasheet specified endurance limit of 100K P/E cycles. This is due to detrapping during the recovery periods between stress events. We can also observe that, although the number of P/E cycles for the most heavily stressed block in EXCH is significantly lower than that in MSNFS, the $\delta V_{th}$ of MSNFS is only slightly higher than that of EXCH. This is because the relationship between $\delta V_{th}$ and the number of P/E cycles is not linear, as discussed in Section 5.1.




(a) P/E cycles experienced over the service life.
The error-bars correspond to the smallest and largest
number of cycles for a block in the SSD.




(b) Change in $\delta V_{th}$ for the block with the highest P/E cycles.
$\Delta V_{th}$=1.7V for SLC Flash.

Figure 6.1: Endurance results for enterprise workloads.

| **Benchmarks** | **Recovery Period (seconds)** | | | |
|---|---|---|---|---|
| | **[1k-5k)** | **[5k-10k)** | **[10k-15k)** | **[15k-20k)** |
| **LM** | 100 | 0 | 0 | 0 |
| **RAD** | 0.001 | 0.0023 | 99.9957 | 0.001 |
| **EXCH** | 0.002 | 99.857 | 0.141 | 0 |
| **MSNFS** | 100 | 0 | 0 | 0 |

Table 6.2: Recovery time distributions. $[X, y)$ indicates recovery periods of duration $t$ where $X \leq t < y$.

The distribution of the recovery periods of the SSD blocks for the workloads over the service life is given in Table 6.2. We can see that most, if not all, SSD blocks in all four workloads experience recovery periods in the order of thousands of seconds. As Table 5.1 indicates, recovery periods of such durations can significantly boost endurance, allowing the blocks to undergo several *millions* of P/E cycles before reaching the endurance limit. The amount of time required for reaching the endurance limit is much longer than the NAND flash retention period. Therefore, endurance is *not* a major flash reliability concern under realistic data center usage scenarios and a much wider array of I/O intensive applications can leverage the performance and power benefits of flash-based SSDs than previously assumed.

# Chapter 7

## Related Work

In this chapter, we review key related work on power and endurance modeling.

**Power Modeling:** Power has been extensively studied by computer architects. [8] provides a framework for analyzing and optimizing the power consumption of microprocessors. [40] provides a detailed modeling of the power consumption of hard disk drives, while [50] has been used to study the power consumption of the memory system. [16] analyzes power from a full system perspective and quantifies the impact of application and operating system on the power behavior of the microprocessor, memory hierarchy and hard disks. Due to lack of a detailed power model for NAND flash memory, existing studies use datasheets like [39] to determine the power consumption. Architects can now use *FlashPower* to study the power consumption of NAND flash memory in detail.

**Charge Trapping/Detrapping in CMOS Transistors:** The generation of interface traps at the $Si/SiO_2$ interface causes a reliability problem for CMOS transistors as well and has been studied in the context of Negative Bias Temperature Instability (NBTI) and Positive Bias Temperature Instability (PBTI). Similar to recovery in FGTs, the interface traps can be removed (detrapped) by applying a suitable logic value as input to the gate of the device and a number of techniques have been proposed to achieve this at runtime [1, 48, 24, 42]. The phenomenon of charge trapping in FGTs has been studied at the transistor level [51, 30, 52] and has been shown to provide compelling endurance benefits. In this thesis, we explore how charge trapping and detrapping affects the endurance of NAND flash memory when exercised by real enterprise workloads.

**Wear-Leveling Techniques for Flash:** A number of wear-leveling techniques have been proposed to balance the wear on flash memory blocks within an SSD to improve endurance and they are discussed in [12]. The proposed techniques include threshold based schemes using per-block erase-counters, techniques that use randomization to choose erase blocks, and those that separate hot and cold blocks when making wear leveling decisions [12]. While these wear-leveling techniques implicitly take into account the impact of charge trapping by counting the number of P/E cycles on a page/block to effect a policy, they do not consider the impact of recovery.

**NAND Flash Power and Endurance Measurements:** There have been recent efforts to understand the characteristics of flash memory by measuring and reverse engineering NAND flash chips [14, 6]. Grupp et al. [14] study the performance, power, reliability of several SLC and MLC NAND flash chips and show that the endurance of these chips tend to be much higher than their datasheet values. The power measurements estimated by *FlashPower* are validated against the chip level measurements studied by [14]. Desnoyers [6] conducted a similar study of the performance and endurance characteristics of several NAND flash memory chips and found the endurance trends to be similar to those reported in [14]. These papers show that the number of P/E cycles that the pages and blocks can sustain is much higher than those given in datasheets. However, these papers do not explain the underlying cause for this trend.

# Chapter 8

# Conclusions and Future Work

Flash memory is used in a wide range of systems varying from consumer electronics to data centers. To support such a diverse range of systems, tools that provide detailed insights into the characteristics of NAND flash memory are required. This thesis describes two tools that model two important characteristics of NAND flash, namely power consumption and endurance. We have presented *FlashPower*, a detailed analytical power model for NAND flash based memory systems and suitable for use in architecture level studies. *FlashPower* models the energy dissipation of flash chips from first principles and is highly parameterized to study a large design space of memory organizations. We have validated *FlashPower* against power measurements from real NAND flash chips and find that our model can accurately estimate the energy dissipation of several real chips. To understand NAND flash endurance, we have developed *FENCE* to model the stress and recovery processes that interact to determine endurance. Using *FENCE*, we show that charge detrapping, which prior architecture and system studies do not consider, can significantly boost endurance. Using a set of real enterprise workload traces, we show that detrapping can allow for orders of magnitude higher number of P/E cycles than those given in datasheets, thereby indicating that SSDs can safely be used in data centers without endurance concerns.

In the future, we plan to extend *FlashPower* to support MLC based NAND flash memory. Similar to *FlashPower*, we plan to validate the endurance model with real chip measurements and we are in the process of building a test board to perform this validation. As mentioned in Section 5.1.3, our model does not capture the impact of temperature on stress and recovery. We plan to

model the impact of this parameter in the future. Another aspect of reliability we plan to model is Bit Error Rate (BER). Prior studies have shown that BERs depend on the age of a flash chip [6]. Modeling the relation between the age of a flash chip and BERs will provide insights into the strength of ECC required for correcting such errors. We also plan to model SILC to factor-in retention. Overall, modeling all these phenomena can provide a comprehensive analysis framework for studying the power and reliability characteristics of NAND flash memory.

# Bibliography

[1] J. Abella, X. Vera, and A. Gonzalez. Penelope: The NBTI-Aware Processor. In *Proceedings of the 40th IEEE/ACM International Symposium on Microarchitecture*, 2007.

[2] N. Agrawal and et al. Design Tradeoffs for SSD Performance. In *Proceedings the USENIX Technical Conference (USENIX)*, June 2008.

[3] A.M. Caulfield and L.M. Grupp and S. Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 217–228, March 2009.

[4] S. Aritome and et al. Reliability issues of flash memory cells. *Proceedings of the IEEE*, 81(5):776–788, May 1993.

[5] A. Birrell, M. Isard, C. Thacker, and T. Wobber. A design for high-performance flash disks. *SIGOPS Operating System Rev.*, 41(2):88–93, 2007.

[6] S. Boboila and P. Desnoyers. Write Endurance in Flash Drives: Measurements and Analysis. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, February 2010.

[7] J.E. Brewer and M. Gill, editors. *Nonvolatile Memory Technologies with Emphasis on Flash*. IEEE Press, 2008.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 83–94, June 2000.

[9] C. Dirik and B. Jacob. The Performance of PC Solid-State Disks (SSDs) as a Function of Bandwidth, Concurrency, Device Architecture, and System Organization. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 279–289, June 2009.

[10] Taehee Cho and et al. A dual-mode NAND flash memory: 1-Gb multilevel and high-performance 512-Mb single-level modes. *Solid-State Circuits, IEEE Journal of*, 36(11):1700–1706, Nov 2001.

[11] P. Desnoyers. Empirical Evaluation of NAND Flash Memory Performance. In *Proceedings of the Workshop on Hot Topics in Storage and File Systems (HotStorage)*, October 2009.

[12] E. Gal and S. Toledo. Algorithms and Data Structures for Flash Memories. *ACM Computing Surveys*, 37(2):138–163, June 2005.

[13] G.R. Ganger, B.L. Worthington, and Y.N. Patt. *The DiskSim Simulation Environment Version 4.0 Reference Manual*. http://www.pdl.cmu.edu/DiskSim/.

[14] L. Grupp and et al. Characterizing flash memory: Anomalies, observations, and applications. In *Microarchitecture, 2009. MICRO-42. 2009 42nd IEEE/ACM International Symposium on*, Nov. 2009.

[15] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 229–240, 2009.

[16] S. Gurumurthi and et al. Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, page 141. IEEE Computer Society, 2002.

[17] S. Gurumurthi, A. Sivasubramaniam, and V. Natarajan. Disk Drive Roadmap from the Thermal Perspective: A Case for Dynamic Thermal Management. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 38–49, June 2005.

[18] Intel X25-E Extreme SATA Solid-State Drive. `http://www.intel.com/design/flash/nand/extreme/index.htm`.

[19] K. Ishida and et al. A 1.8V 30nJ adaptive program-voltage (20V) generator for 3D-integrated NAND flash SSD. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 238–239,239a, Feb. 2009.

[20] Process Integration and Device Structures, ITRS 2007 Edition. http://www.itrs.net/Links/2007ITRS/2007_Chapters/2007_PIDS.pdf.

[21] T. Kgil, D. Roberts, and T. Mudge. Improving NAND Flash Based Disk Caches. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 327–338, June 2008.

[22] J.K. Kim and et al. A 120-$mm^2$ 64-Mb NAND flash memory achieving 180 ns/Byte effective program speed. *Solid-State Circuits, IEEE Journal of*, 32(5):670–680, May 1997.

[23] J.K. Kim and et al. A PRAM and NAND flash hybrid architecture for high-performance embedded storage subsystems. In *EMSOFT '08: Proceedings of the 8th ACM international conference on Embedded software*, pages 31–40, New York, NY, USA, 2008. ACM.

[24] S.V. Kumar, C.H. Kim, and S.S. Sapatnekar. Impact of NBTI on SRAM Read Stability and Design for Reliability. In *Proceedings of the International Symposium on Quality Electronic Design*, 2006.

[25] J. Lee and et al. A 1.8V NAND Flash Memory for Mass Storage Applications. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 290–494, February 2003.

[26] J. Lee and et al. A 90-nm CMOS 1.8-V 2-Gb NAND flash memory for mass storage applications. *Solid-State Circuits, IEEE Journal of*, 38(11):1934–1942, Nov 2003.

[27] M. Lenzlinger and E.H. Snow. Fowler-Nordheim tunneling into thermally grown Si$O_2$. *Electron Devices, IEEE Transactions on*, 15(9):686–686, Sep 1968.

[28] Y. Manabe and et al. Detailed Observation of Small Leak Current in Flash Memories with Thin Tunnel Oxides. In *Proceedings of the International Conference on Microelectronic Test Structures (ICMTS)*, pages 95–99, March 1998.

[29] N. Mielke and et al. Flash eeprom threshold instabilities due to charge trapping during program/erase cycling. *Device and Materials Reliability, IEEE Transactions on*, 4(3):335–344, Sept. 2004.

[30] N. Mielke and et al. Recovery effects in the distributed cycling of flash memories. In *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, pages 29–35, March 2006.

[31] Neal R. Mielke. Intel Corporation, October 2009. Private Correspondence.

[32] H. Nakamura and et al. A 125mm$^2$ 1Gb NAND Flash Memory with 10 MB/s Program Throughput. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 106–107, February 2002.

[33] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, February 2008.

[34] D. Narayanan and et al. Migrating Enterprise Storage to SSDs: Analysis of Tradeoffs. In *Proceedings of EuroSys 2009*, March 2009.

[35] Y. Park and D.K. Schroder. Degradation of thin tunnel gate oxide under constant fowlernordheim current stress for a flash eeprom. In *IEEE Transactions on Electron Devices*, 1998.

[36] S. Kavalanekar and et al. Characterization of storage workload traces from production Windows servers. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, pages 119–128, October 2008.

[37] S. Lee and et al. FlexFS: A Flexible Flash File System for MLC NAND Flash Memory. In *Proceedings of the USENIX Annual Technical Conference (USENIX)*, June 2009.

[38] Koji Sakui. Intel Corporation/Tohoku University, May 2009. Private Correspondence.

[39] Samsung corporation. K9XXG08XXM flash memory specification.

[40] S. Sankar, Y. Zhang, S. Gurumurthi, and M.R. Stan. Sensitivity-Based Optimization of Disk Architecture. *IEEE Transactions on Computers*, 58(1):69–81, January 2009.

[41] R.K. Sharma and et al. Balance of power: dynamic thermal management for internet data centers. *Internet Computing, IEEE*, 9(1):42–49, Jan.-Feb. 2005.

[42] J. Shin, V. Zyuban, P. Bose, and T. Pinkston. A Proactive Wearout Recovery Approach of Exploiting Microarchitectural Redundancy to Extend Cache SRAM Lifetime. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 353–362, June 2008.

[43] SNIA IOTTA Trace Repository. http://iotta.snia.org/.

[44] M.R. Stan and W.P. Burleson. Bus-Invert Coding for Low-Power I/O. *IEEE Transactions on Very Large Scale Integration Systems*, 3(1):49–58, March 1995.

[45] K.D. Suh and et al. A 3.3V 32 Mb nand flash memory with incremental step pulse programming scheme. *Solid-State Circuits, IEEE Journal of*, 30(11):1149–1156, Nov 1995.

[46] T. Tanaka and et al. A quick intelligent page-programming architecture and a shielded bitline sensing method for 3V-only nand flash memory. *Solid-State Circuits, IEEE Journal of*, 29(11):1366–1373, Nov 1994.

[47] Application Report: MSP430 Flash Memory Characteristics. http://focus.ti.com/lit/an/slaa334a/slaa334a.pdf.

[48] A. Tiwari and J. Torrellas. Facelift: Hiding and Slowing Down Aging in Multicores. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, November 2008.

[49] David Tawei Wang. *Modern DRAM Memory Systems: Performance analysis and a high performance, power-constrained DRAM scheduling algorithm*. PhD thesis, University of Maryland, College Park, 2005.

[50] S.J.E. Wilton and N.P. Jouppi. Cacti: an enhanced cache access and cycle time model. *Solid-State Circuits, IEEE Journal of*, 31(5):677–688, May 1996.

[51] R. Yamada and et al. A novel analysis method of threshold voltage shift due to detrap in a multi-level flash memory. In *Symposium on VLSI Technology, Digest of Technical Papers*, 2001.

[52] H. Yang and et al. Reliability issues and models of sub-90nm NAND flash memory cells. In *International Conference on Solid-State and Integrated Circuit Technology*, 2006.