Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs

Qi Wu, Guiqiang Dong, and Tong Zhang
ECSE Department, Rensselaer Polytechnic Institute (RPI), USA

***Abstract*—This paper proposes a self-healing solid-state drive (SSD) design strategy that exploits heat-accelerated recovery of NAND flash memory cell wear-out to improve SSD lifetime. The key is to make each NAND flash memory chip self-healable by stacking an extra heater die, and to employ system-level redundancy to ensure SSD data storage integrity when one memory chip is being self-heated for memory cell wear-out recovery. Based upon detailed thermal modeling and memory cell device modeling, we carried out simulations and performed detailed analysis. The results show that SSD lifetime can be improved by over five times at reasonable performance and energy consumption overhead.**

I. Introduction

The steady bit cost reduction of NAND flash memory now makes it economically viable to implement solid-state drive (SSD) using NAND flash memory. Nevertheless, as the technology continues to scale down, the achievable program/erase (P/E) cycle of NAND flash memory continuously degrades. It is the most critical obstacle for a wide adoption of SSDs, particularly for enterprise class applications. P/E cycling wears out NAND flash memory by creating more oxide and interface traps in memory cells. It has been known that interface traps recover with time and the recovery speed is faster under higher temperature [1], [2]. Such interface trap recovery has been conventionally treated as a reliability problem leading to memory retention limit.

We propose to use interface trap recovery as an advantageous leverage to enable self-healing SSDs, which can survive much more P/E cycles. In self-healing SSDs, each NAND flash memory chip stacks one or multiple NAND flash memory die(s) with a self-heater die containing arrays of polysilicon resistors. Each NAND flash memory chip can be self-heated to certain high temperature for recovering most interface traps during a reasonable amount of time. Since most SSDs contain multiple NAND flash memory chips and one dedicated controller, we can add system-level redundancy to ensure data storage integrity, i.e., the SSD contains at least one backup (or spare) memory chip, and before one memory chip is self-heated for accelerating interface trap recovery, we always copy its content to one backup chip. In the run time, all the memory chips are self-heated alternatively and the SSD controller can dynamically determine which chip is used as backup chip. Since all the memory chips on the same SSD channel share a common bus, data backup traffic will interfere with normal I/O requests and hence degrade SSD response time. To alleviate this issue, we propose a data backup scheduling strategy. To estimate the potential effectiveness, we developed a NAND flash memory cell model, a chip-level thermal model and a modified SSD system simulator [3], [4], based on which we carried out detailed simulations and theoretical analysis on the interface trap recovery effects. The results show that such self-healing SSD design strategy can potentially increase the SSD lifetime by over five times with reasonable performance and energy consumption overhead.

II. Self-Healing SSD Architecture

The proposed self-healing SSD design strategy aims at on-the-fly recycling the NAND flash memory chips by exploiting the heat-accelerated interface trap recovery phenomenon. To realize this, each memory chip should be able to self-boost its temperature to a sufficiently high value. For this purpose we propose to stack one heater die with the existing one or multiple memory dies in each NAND flash memory chip. The heater die can simply contain arrays of polysilicon resistors for generating heat to boost the temperature within the chip package. We note that, although memory P/E cycle induces traps in both bulk oxide and Si-SiO_2 interface, it is generally believed that only interface traps can noticeably recover. When a memory chip is periodically self-heated over the entire lifetime, the interface trap density will oscillate, while the bulk oxide trap density will monotonically increase. As a result, a NAND flash memory chip can be self-heated up to a certain number.

Since the memory chip being heated for trap recovery cannot store user data, SSDs must contain system-level redundancy (i.e., extra memory chips) to prevent user data loss, and the SSD controller should determine when and which memory chip is self-heated. Therefore, as illustrated in Fig. 1, a self-healing SSD contains at least one extra NAND flash memory chip as the backup which does not occupy any address space. Before one NAND flash memory chip is self-heated, the SSD controller must copy its content to a backup chip. In the run time, all the NAND flash chips are self-heated alternatively and the SSD controller can dynamically determine which chip is used as backup chip. This can be illustrated in Fig. 1, where the sick chip refers to the memory chip that is most worn-out and must be self-heated to prevent user data loss, and the others are called healthy chip and the chip self-heated most recently is used as the backup chip. The SSD controller can use certain parameters (e.g., P/E cycle or worst-case memory read bit error rate (BER)) to determine which memory chip is the sick chip, or the controller can use un-balanced wear-

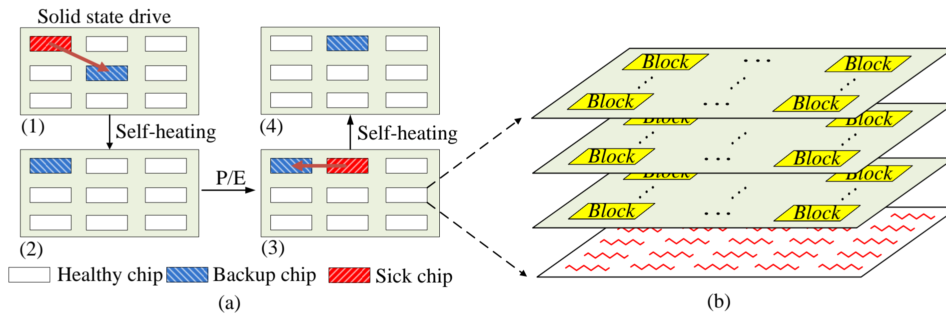

Fig. 1. Illustration of self-healing solid state drive: (a) self-healing operation process, and (b) each NAND flash memory chip with stacked self-heater.

leveling to intentionally make one chip wear out more quickly than the others.

When the content of the sick chip is migrated to the backup chip, the address of the data is still the same since backup chip does not occupy any address space. Therefore, the page table does not need to be changed. However, the decoding of chip select signals needs to be changed since the sick chip becomes the backup chip after data backup process. To minimize the effect of data backup traffic on SSD I/O request performance, we propose to enhance SSD controller scheduling as follows: (1) SSD controller should always try to issue the data backup command when SSD is idle; (2) Normal I/O requests have higher priority than data backup and can interrupt data backup operations; (3) If I/O read requests involve data in the sick chip and the data have already been copied to the backup chip, those read requests should be issued to the backup chip directly; (4) If I/O write requests target at the sick chip, they should be re-directed to the backup chip.

III. Simulation Setup

A. Thermal modeling

Fig. 2 shows the overall structure of the self-healing NAND flash memory chip in our thermal simulation. One side of the

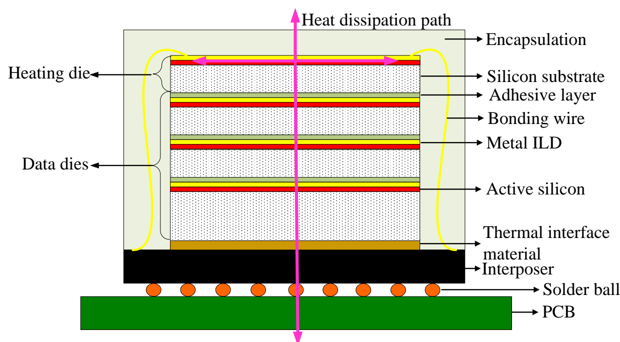

Fig. 2. Structure of the 3D self-healing flash chip in thermal simulation.

chip is encapsulation and the other side is thermal interface, interposer, solder ball, and PCB. In the middle of the chip are one or multiple NAND flash memory dies and one heater die. Every die is composed of metal layers, active silicon and silicon substrate. There is also an adhesive layer between every two adjacent dies. The thickness and thermal conductance of every layer are shown in Table I. Heat generated by the

TABLE I
Chip layer configurations.

Layer	Thermal conductivity	Thickness
Encapsulation	0.453 W/(mk)	1.0 mm
Silicon substrate	100 W/(mK)	50 μm
Adhesive material	4 W/(mk)	4 μm
Metal ILD	200 W/(mk)	8 μm
Active silicon	100 W/(mk)	2 μm
Thermal interface material	4 W/(mk)	20 μm
Interposer	2 W/(mk)	0.4 mm
Solder balls	16.7 W/(mk)	0.94 mm
PCB	3 W/(mk)	2 mm

heater die on the top can be dissipated either from the top going through the encapsulation to the air or from the bottom going through all the flash memory dies, thermal interface, interposer, solder ball, and PCB to the air. The latter path is a major heat dissipation path since PCB has much less convection resistance than the encapsulation. We use the HotSpot thermal simulator [5] to carry out very fast yet reasonably accurate thermal simulation based upon equivalent circuits of thermal resistances and capacitances. However, existing Hotspot simulator does not support the low power/cost chips without heat spreader/sink and does not support the secondary heat dissipation path in stacked chip thermal simulation. In this work, we further enhanced this simulation tool to support these two features, and the main configuration parameters used in our simulation are listed in Table II.

TABLE II
HotSpot simulation parameters.

Parameter	Value
Chip package	100 ball PBGA (12mm x 18mm x 1.4mm)
Die size	9.28mm x 12.96mm
HotSpot grid size	64 x 64 (0.145mm x 0.203mm per grid)
Ambient temperature	45 °C
PCB convection resistance	50 K/W
Encapsulation convection resistance	200 K/W

B. NAND Flash Memory Cell Modeling

We also developed a memory cell model to facilitate the evaluation. It quantitatively models the interface and oxide trap generation, and temperature-dependent interface trap recovery. The interface and oxide trap generation grows with P/E cycling in a power-law manner, i.e., let ΔN_{trap} and N_{cyc} denote the trap generation and P/E cycle number, we have $\Delta N_{trap} \propto N_{cyc}^{\alpha}$. According to [6], we set the exponent α for interface and oxide trap generation as 0.62 and 0.30, respectively. Regarding interface trap recovery, we set [6]:

$$dN_{it}/N_{it} = -k \cdot dt, \tag{1}$$

where N_{it} is the interface trap density, t is the self-heating time, k is a function of temperature. It clearly suggests that the interface trap density N_{it} scales down with e^{-kt}. Based on the Arrhenius approximation, k is obtained as $k_0 \cdot e^{-E_a/RT}$, where k_0 is the intrinsic rate constant, E_a is the activation energy of hydrogen on Si-H bond and its value is 0.52eV, and R is the gas constant.

This model further explicitly captures the effects of random-telegraph noise (RTN), retention noise, and cell-to-cell interference. RTN causes random fluctuation of memory cell threshold voltage, where the fluctuation magnitude is subject to exponential decay. Hence, we model the probability density function of RTN-induced threshold voltage fluctuation as a symmetric exponential function [7]: $p_r(x) = \frac{1}{2\lambda_r} e^{-\frac{|x|}{\lambda_r}}$. Since RTN is induced by electron capture and emission events at interface traps, we set the mean of RTN follows $A_r \cdot N_{cyc}^{0.62}$, where A_r is set as 1.80e-4.

Threshold voltage reduction due to retention noise is approximately modeled as a Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$. The mean value of retention shift is proportional to both interface trap generation and oxide trap generation, i.e., $\mu_d = A_t \cdot N_{cyc}^{0.62} + B_t \cdot N_{cyc}^{0.3}$, where we set A_t=7.0e-4, B_t=4.76e-3, and $\sigma_d = 0.3\mu_d$ in our simulation. In addition, to approximately model the dependence of retention shift on the programmed threshold voltage, i.e., a higher programmed threshold voltage tends to result in a larger threshold voltage shift during retention, we set that the retention noise approximately scales with $K_s(x - x_0)$, where x is the initial threshold voltage, and x_0 and K_s are constants.

Threshold voltage shift of a victim cell caused by cell-to-cell interference can be estimated as $F = \sum_k (\Delta V_t^{(k)} \cdot \gamma^{(k)})$, where $\Delta V_t^{(k)}$ represents the threshold voltage shift of one interfering cell which is programmed after the victim cell, and the coupling ratio $\gamma^{(k)}$ is defined as $\gamma^{(k)} = \frac{C^{(k)}}{C_{total}}$, where $C^{(k)}$ is the parasitic capacitance between the interfering cell and the victim cell, and C_{total} is the total capacitance of the victim cell. This study assumes that NAND flash memory employs the all-bit-line structure, which suffers less interference by programming all the cells on one wordline at the same time. The coupling ratio between two vertically and diagonally adjacent cells is set as 0.096 and 0.0072, respectively.

C. SSD System Performance Modeling

Our SSD system performance simulation is based on the DiskSim Simulator [3] with SSD model patch [4]. We use 14 I/O traces including Iozone and Postmark [4], Finance1, Finance2, WebSearch1, WebSearch2, and WebSearch3 from [8], and Trace 1-7 from [9]. In our simulation, each chip contains 2 dies that share an 8-bit I/O bus and a number of common control signals, and each die contain 2 planes and each plane contains 2048 blocks. Each block contains 64 pages, each of which consists of 8 sectors (512 Byte). There are 2 channels (gangs) in the simulated SSD. Each channel harbors 17 flash chips including one backup chip. The backup chip on each channel only backups the data of the flash chips on the same channel. We also configure the simulator to support data stripping over 2 channels. Following the version 2.0 of the Open NAND Flash Interface (ONFI) [10], we set the bus bandwidth as 133MB/s. We set flash memory read access time as 50μs and program time as 600μs [11]. The data backup operation can be divided into 4 stages: (1) read data from sick chip NAND flash cells to page register, (2) transfer data from sick chip page register to the controller for ECC decoding, (3) transfer data from the controller to backup chip page register, and (4) write data from backup chip page register to NAND flash celss. In our simulation, the backup operations start only when SSD is idle. The four stages of backup operation of different planes can be pipelined and operated in parallel [4]. There are 2 situations under which the I/O request is delayed due to the data backup operation. First, at least one data backup operation is in stage (2) or (3) when the I/O read request comes. This is because the bus is occupied by data backup operation in backup stage (2) and (3) and backup operation can only stop at certain granularity boundary. Second, the I/O read request is accessing the un-backup data in the sick chip and there are un-transferred data in the page register of the sick chip. Under other situations, the I/O request will not be delayed.

IV. SIMULATION RESULTS

Fig. 3 shows the calculated interface traps recovery under different self-heating temperature and time based upon Eq. (1). The results clearly show a strong dependence of interface trap recovery on the self-heating temperature. Using the parameters listed in Table II, we carried out Hotspot simulations to estimate the power consumption required to reach different temperatures. As shown in Fig. 3, different temperature results in different interface trap recovery speed. Hence, by setting the target of recovering 80% of total interface traps, different self-heating temperature corresponds to different required heating time and may consume different total energy consumption. Fig. 4 shows the simulated power and energy consumption results for different heating temperatures. The power consumption of the stacked heater die almost linearly increases from 1.5W to 5.1W in order to increase the temperature from 110°C to 250°C. Meanwhile, as shown in Fig. 3 the time for interface trap recovery drops almost exponentially when the temperature increases from 110°C to 250°C. Therefore, as shown in Fig. 4

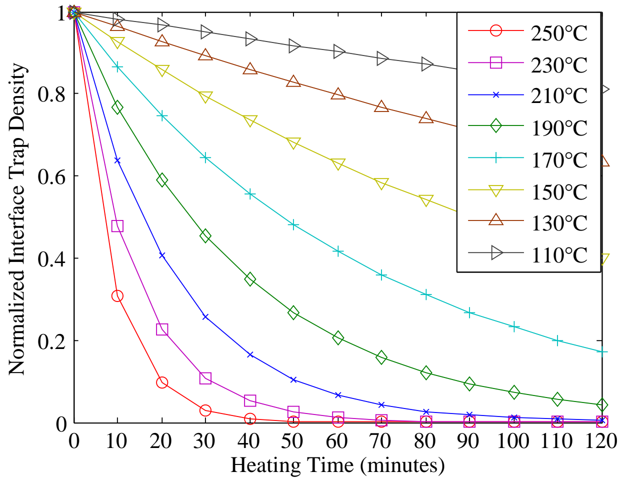

Fig. 3. Interface trap recovery as a function of heating time under different temperature.

the total self-heating energy consumption almost exponentially reduces as the temperature increases. Therefore, from the heating efficiency perspective, we may want to increase the temperature as high as possible. However, the melting points for the solder balls are ranged from 210∼250 °C [12]. Furthermore, different materials have different thermal expansion coefficients. Hence heating up the chip beyond a certain threshold value can result in package body warpage and some chip may get damage, e.g., adjacent solder ball joint short, metal wire open, etc. In this study, we choose 200°C as the target self-heating temperature.

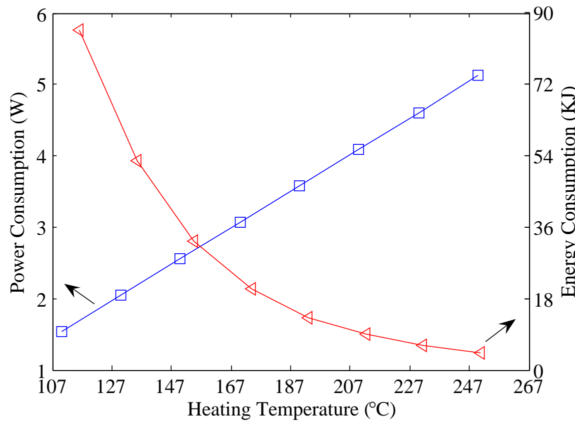

Fig. 4. Power and energy consumption of the self-heating die under different heating temperature when targeting at 80% interface trap recovery.

Fig. 5 shows the total achievable P/E cycles of a self-healing 2bits/cell NAND flash memory chip based upon the above memory cell models. Note that we conservatively assume that only 80% of the interface traps are generated since the previous self-healing can recover in present self-healing, and the remaining interface traps are permanently un-recoverable and will accumulate. The interface trap can also recover under normal room temperature with much slower speed, compared to being heated under much higher temperature. Therefore, the same SSD can have different P/E cycling endurance numbers under different write intensities. Assuming the allowable worst-case memory read raw BER of 2.04e-3 under 10-year retention limit and 25MB/s average write volume, P/E cycle can be about 3,000. When using self-heating, we set that self-heating is triggered once the estimated worst-case memory read raw BER reaches 1.50e-3. Clearly, due to the residual interface trap accumulation and un-recoverable oxide traps, the achievable number of P/E cycles between two consecutive self-heating operations monotonically reduces as in Fig. 5. If it reduces to below 200, the SSD self-healing operation will stop and the SSD controller claims the end of SSD lifetime.

According to the results in Fig. 3, it needs up to 35 minutes for 80% interface traps to recover under 200°C. Assuming there are 34 flash memory chips in the self-healing SSD (including two backup chips) and cooling down time is 3 times longer than the self-heating time, we need total $35 \times 4 \times 32/60 = 75$ hours to self-heat the 32 memory chips once, which should be easily justified since the lifetime of SSDs typically is a few years. According to the results shown in Fig. 5, P/E cycling endurance can increase up to about 17,400 P/E cycles, which represents almost 6x improvement compared with the original 3,000 P/E cycling endurance.

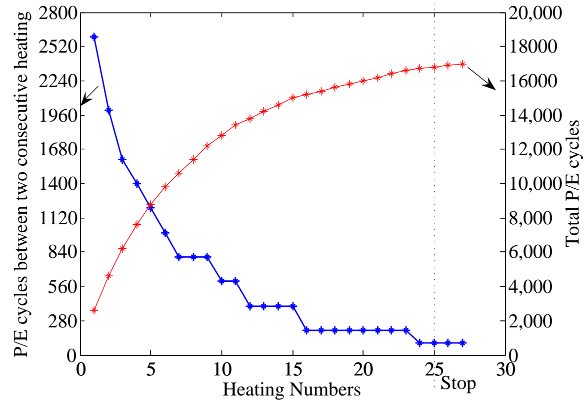

Fig. 5. Total P/E cycles and P/E cycles between two consecutive heating as self-heating increasing.

Fig. 6 shows the average response time increase over different traces when data backup is invoked. Among all traces, Financial2, WebSearch1, WebSearch2, and WebSearch3 entail much higher response time increment than other I/O traces do. This is because these four I/O traces mainly contain relatively small I/O requests. The delay induced by data backup is more significant for them. For each single I/O trace, the average response time increases with data backup interruption granularity increasing from 1 byte to 4096 bytes. In fact, there is a tradeoff between average response time increase and data backup speed. A finer interruption granularity will increase average response time but reduce the backup time at the same time. However, according to our simulation, the variance in the backup speed is less than 1%. This is mainly because the idle time in these I/O traces are relatively large,

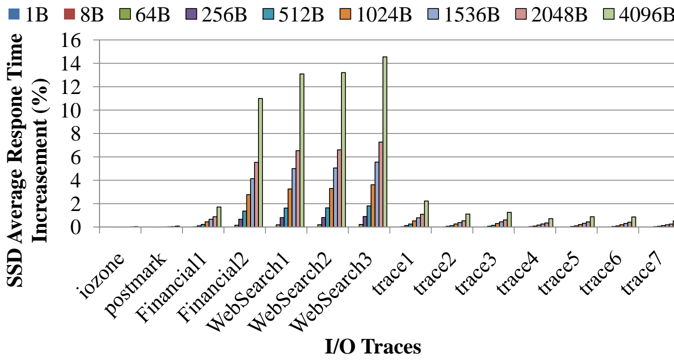

Fig. 6. SSD average response time increment over different I/O traces when data backup is invoked. Data backup operation interruption granularity varies from 1B to 4096B.

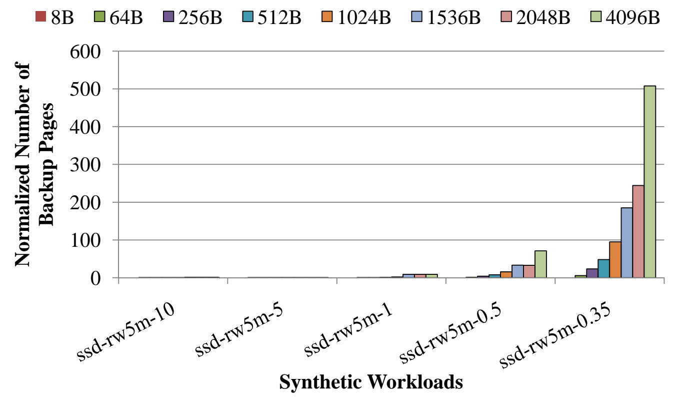

Fig. 8. Number of pages that can be copied to the backup chip throughout different synthetic workloads when data backup is invoked, which are normalized to the scenario with interruption granularity of 1B.

i.e., the workloads are not intensive enough to show this tradeoff. Therefore, we use four synthetic workloads generated in DiskSim to illustrate this tradeoff. These four synthetic traces are all composed of 5 million random read and write I/O operations with the mean of inter-arrival time varying from 10ms to 0.35ms. Fig. 7 shows the SSD average response time increment over different I/O traces when data backup is invoked, and Fig. 8 shows the number of pages that can be copied to the backup chip during these traces, which is normalized to the scenario with interruption granularity of 1B. Results suggest that given very intensive workload we should dynamically change the data backup interruption granularity according to the characteristic of the I/O traces to control both average response time increment and backup speed.

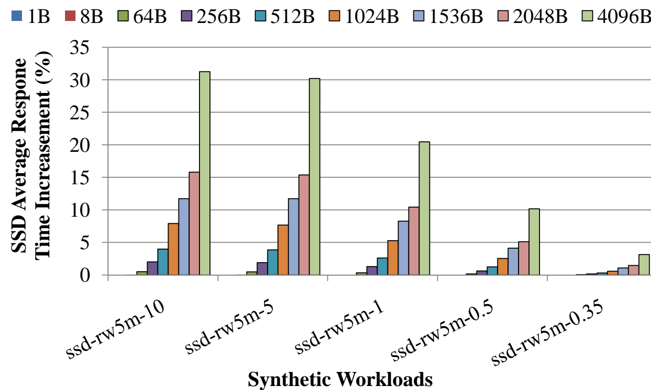

Fig. 7. SSD average response time increment over different synthetic workloads when data backup is invoked. Data backup operation interruption granularity varies from 1B to 4096B.

V. Conclusion

This paper presents a self-healing SSD design strategy to improve the SSD lifetime by leveraging the inherent interface trap recovery phenomenon of NAND flash memory cells. The basic building element is self-healing NAND flash memory chip that stacks a separate heater die with NAND flash memory die(s) in the same chip package. The architecture of self-healing SSDs and implication to SSD controller scheduling are discussed. Simulation and analysis results based on detailed thermal and flash memory cell models suggest a potential of achieving over five times improvement of SSD lifetime at reasonable energy consumption and speed performance cost.

References

[1] J.D. Lee, J.H. Choi, D. Park, and K. Kim, "Effects of interface trap generation and annihilation on the data retention characteristics of flash memory cells," *IEEE Transactions on Device and Materials Reliability*, vol. 4, pp. 110 – 117, March 2004.

[2] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 3, pp. 335–344, 2004.

[3] *The DiskSim Simulation Environment Version 3.0 Reference Manual.0*, http://www.pdl.cmu.edu/DiskSim/.

[4] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy, "Design tradeoffs for SSD performance," in *USENIX Annual Technical Conference*, Boston, MA, June 2008, pp. 57–70.

[5] K. Skadron et al., "Temperature-Aware Microarchitecture," in *Proceedings of the 35th ACM/IEEE International Symposium on Computer Architecture,*, 2003, pp. 2–13.

[6] H. Yang et al., "Reliability issues and models of sub-90nm NAND Flash memory cells," in *International conference on Solid-State and Integrated Circuit Technology*, Oct. 2006, pp. 760–762.

[7] C.M. Compagnoni, M. Ghidotti, A.L. Lacaita, A.S. Spinelli, and A. Visconti, "Random Telegraph Noise Effect on the Programmed Threshold-Voltage Distribution of Flash Memories," *IEEE electron device letters*, vol. 30, no. 9, 2009.

[8] *SPC Trace File Format Specification*, http://traces.cs.umass.edu/index.php/Storage/Storage.

[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," *SIGARCH Comput. Archit. News*, vol. 37, pp. 279–289, 2009.

[10] *Open NAND Flash Interface Speci?cation*, http://onfi.org/specifications/.

[11] R. Micheloni, M. Picca, S. Amato, H. Schwalm, M. Scheppler, and S. Commodaro, "Non-volatile memories for removable media," *Proceedings of the IEEE*, vol. 97, pp. 148–160, Jan. 2009.

[12] B.T. Vaccaro, R.L. Shook, and D.L. Gerlach, "The impact of Lead-Free reflow temperatures on the moisture sensitivity performance of plastic surface mount packages," in *International Workshop on Thermal Investigations of ICs and Systems*, Oct. 2009, pp. 113–116.