Addressing Fast-Detrapping for Reliable 3D NAND Flash Design

Mustafa M. Shihab[1], Jie Zhang[2], Myoungsoo Jung[3] and Mahmut Kandemir[4]
[1]Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson 75080, USA
[2]School of Integrated Technology, Yonsei University, Incheon, South Korea
[3]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea
[4]Dept. of CSE, Pennsylvania State University, University Park, Pennsylvania, USA
mustafa.shihab@utdallas.edu, jie@yonsei.ac.kr, mj@camelab.org, and kandemir@cse.psu.edu

1 INTRODUCTION

In order to resolve the scalability challenges [3] of the conventional 2D architecture, the NAND flash community has adopted designs that vertically stack cells and expand storage capacity by constructing a 3D array [4, 11]. Unlike 2D NAND, most 3D NAND designs replace conventional *floating-gate* (FG) cells with *charge-trap* (CT) cells for simplifying the vertical fabrication process [9]. Unfortunately, while 2D NAND starts to suffer from high bit error rate (BER) near the end of its retention period, 3D NAND can experience ~70% of the peak BER only months after a program, as shown in Figure 1a. This happens because, 3D NAND encounters a drift in threshold voltage (V_{Th}) relatively soon after a write (program) operation – a phenomenon known as *fast-drift* [1, 6]. Consequently, resolving fast-drift can become an immediate concern for CT-flash based 3D-NAND in order to avoid data loss.

Since the fast-drift phenomenon is unique to CT-flash, the existing NAND flash retention improvement techniques are ineffective against it. Furthermore, run-of-the-mill error-correcting code (ECC) solutions are also infeasible as the energy and latency overheads of such schemes increase super-linearly with error rate. However, repeated in-place programming on CT-flash cells can refill the depleted charge and gradually diminish the impact of fast-drift [7]. As shown in Figure 1b, our array-level simulation results confirm that three extra charge-refill operations after a write can slow-down fast-drift and ensure storage-class data retention. Unfortunately, we observed that such refill operations exceedingly amplify the overheads of each program operation. Therefore, naively scheduling refill operations in 3D NAND can render it unattractive for high-performance and low-power applications.

In this work, we propose ReveNAND – a set of novel measures to counter fast-drift and the resulting errors in 3D NAND. We first present the first analytic model for fast-drift in CT-flash, which can uncover and explore the impact of fast-drift. We then propose our elastic read reference scheme *(ERR)* that can dynamically adjust the V_{Ref} to correctly read data from a fast-drift affected 3D NAND. Next, we introduce *hitch-hike*, a modified page organization for 3D NAND blocks, that utilizes dedicated pages to store error correction bits for the pages that are expected to face the highest number of errors. Finally, we propose a novel reinforcement-learning based intelligent charge-refill scheme *(iRefill)* to diminish the impact of fast-drift with minimum number of refill operations. Our experiments show that, with our proposed counter-measures, ReveNAND can reduce errors from fast-drift by 87%, on an average. As a result, compared to the state-of-the-art, it can lower the ECC latency and energy overheads by 13X and 10X, respectively.

2 FAST-DETRAPPING AND V_{Th} DRIFT

The charge strorage layer (CSL) in CT-flash in 3D NAND is an insulator. Consequently, during a program operation a large fraction of the electrons are shallowly trapped along the tunnel oxide-CSL boundary, as shown in Figure 2a. These shallow-trapped electrons can escape the CSL soon after a program completes, and cause the V_{Th} to drift [1, 6]. As shown in Figure 2b, this V_{Th} drift can eventually spread beyond the threshold reference voltage V_{Ref} and generate error.

Analytic model. To the best of our knowledge, [1] remains the only publicly-available work reporting the detailed behavior of fast-drift phenomenon in the CT-flash cells. Motivated by their empirical data, we analytically characterized the relation between fast-drift and the parameters that critically affect it. By considering all the parameters' cumulative influence on fast-drift, and using the empirical data for obtaining the corresponding fitting constants, we modeled fast-drift as:

$$\Delta V_{Th}(t) = -[log(t) + \alpha][\frac{V_{Th,Init}}{\beta} + \theta\Delta T + \frac{\delta}{t_{buff-ox}}][\frac{1}{R+1}] \quad (1)$$

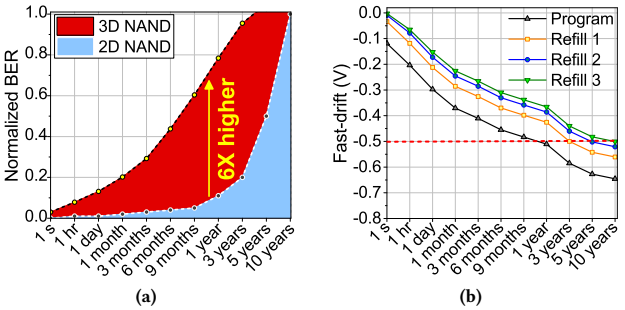
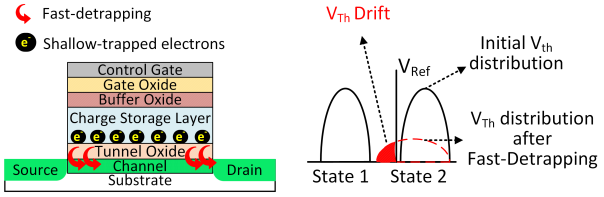


Figure 1: Fast-drift increases error in 3D NAND flash (a), but repeated charge-refills can minimize the adverse impact (b).



(a) Electron detrapping. **(b) V_{Th} drift.**

Figure 2: The fast-drift phenomenon.

where ΔV_{Th} is fast-drift, t is the elapsed time after a write, $V_{Th,Init}$ is the initially programmed V_{Th}, ΔT is the operating temperature (i.e., 20°C), $t_{buff-ox}$ is the buffer-oxide thickness, and R is the refill count. In addition, α, β, θ and δ are the fitting constants with the values derived from [1] as 5, 368, 2.1x10^{-4}, and 47.5x10^{-3}, respectively.

However, compared to the 2D architecture, 3D NAND can allow a significantly higher number of electrons to be shallow-trapped. Specifically, adding layers in a 3D-NAND string can be considered as *linear* stacking of slices of critical surface that can host shallow-trapped charges. Prior works have also reported that, for a fixed programming voltage, fast-drift increases linearly [2]. This is because, in a 3D NAND string, the amount of fast-drift experienced by a layer increases *only with addition of a neighboring layer, and is independent of other layers.* Carefully considering these factors, we assume a linear correlation between fast-drift and the stacked layers in our proposed model. For a P% increase in fast-drift for each stacked layer in 3D NAND, the total fast-drift in a 3D NAND design with n layers can be expressed as:

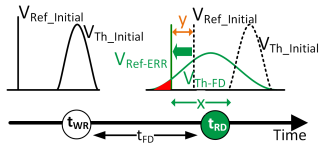
$$\Delta V_{Th-3D} = (1 + \frac{P}{100}(n-1))\Delta V_{Th-Cell} \quad (2)$$

where $\Delta V_{Th-Cell}$ is the fast-drift for a single cell. Equation 2 indicates that, the stacked design can significantly aggravate errors from fast-drift in 3D NAND. It is worth noting that, while our model is based on the TCAT design, it can be applied to any CT-flash based 3D-NANDs.

3 COUNTER MEASURES FOR FAST-DRIFT

3.1 Elastic Read Reference (ERR)

Fast-drift varies with the elapsed time between writing a page and reading it back. If the corresponding V_{Ref} is adjusted proportionally, we can correctly read the affected page. Motivated by this, we propose to maintain a bin of V_{Ref}s, and dynamically assign a V_{Ref} for each read operation based on when that page was last written. Figure 3 illustrates the impact of our proposed ERR scheme, and it operates as follows: 1) The flash controller marks the time of a read request as the *read-time*

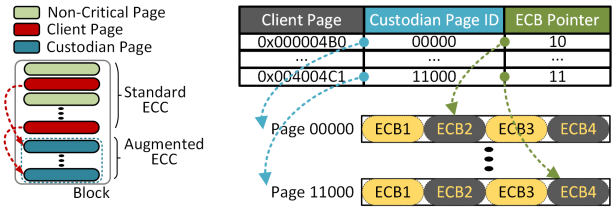


(a) Elastic V_{Ref} dynamically adopts with fast-drift

Time-bin for Est. Fast-drift (days)	ERR Assigned V_{Ref-FD} (V) [50°C]	$\|y\| = V_{Ref-FD} - V_{Ref-Init}$
$t_{FD} < 0.01$	$\|x\| = 0.425$	0.000
$0.01 < t_{FD} < 1$	$\|x\| = 0.45$	0.025
$1 < t_{FD} < 100$	$\|x\| = 0.475$	0.050
$t_{FD} > 100$	$\|x\| = 0.5$	0.075

(b) ERR time-bins and corresponding V_{Ref-FD}

Figure 3: Countering fast-drift with ERR.



(a) (b)

Figure 4: Our hitch-hike scheme: block organization (a), and ECB assignments to the custodian pages (b).

(t_{RD}). 2) The controller then uses the time-stamp for the latest write on that page as the *write-time* (t_{WR}). 3) The effective elapsed time for fast-drift is, *fast-drift time* (t_{FD}) = t_{RD} - t_{WR}. 4) The amount of fast-drift to be experienced by the target page is calculated using our analytic model and a suitable V_{Ref-FD} is assigned accordingly. The table in Figure 3b summarizes the *time-bins* used in our evaluation, and the corresponding V_{Ref-FD} values expressed in terms of their absolute difference ($|x|$) from the initially programmed threshold voltage ($V_{Th\_Initial}$). The parameter $|y|$ is the difference between the ERR assigned V_{Ref-FD} and the $V_{Ref-Initial}$, where the latter is the conventional static V_{Ref} value.

3.2 Hitch-hike: Risk based Prioritization

Detaching the physical mapping of data from the physical mapping of its ECC information can reduce the ECC overhead [12]. Inspired by this, we propose a scheme that can provide an alternate (and stronger) ECC to the most error-prone 3D NAND flash pages by placing their ECBs in a pool of predesignated free pages (hence the name hitch-hike). In this scheme, as shown in Figure 4, most of the pages in a block are available for regular data storage, and are encoded/decoded with the regular ECC module. However, the hitch-hike controller designates a small fraction of the pages as *custodian* pages, and sets them aside for storing the ECB data. When a page retains data for a prolonged period and is expected to be vulnerable to fast-drift, the hitch-hike controller marks them as *client* pages. These critical client pages are then read by the memory controller in the background, and are encoded using an augmented ECC codec. The ECB for this enhanced ECC encoding is stored in a custodian page. When a read is assigned for that client page, the controller accesses *both* the client page and its corresponding custodian page, and decodes the data using the stored ECB.

3.3 iRefill: Affordable Data Retention

Our iRefill scheme can further minimize fast-drift in 3D NAND through optimally scheduled "refill" operations. Specifically, iRefill leverages reinforcement-learning [10] to reduce the number of refills, which in turn can allow 3D NAND to attain storage-class retention with minimum overhead. The iRefill controller collects the state and reward information from the 3D NAND, and assigns an action for the next state. The state functions include – current refill count, elapsed time since last write/refill, and current BER. The action function is either assigning a refill operation or, continuing with the regular operations. While the *immediate reward* for iRefill is to maintain the BER permitted by the ECC scheme, the *long-term reward* is to minimize the refill frequency and maximize I/O throughput. iRefill schedules the refill operations at a block-level granularity to minimize the potential resource overheads. If regular I/O occurs while refilling a block, iRefill *interleaves* refill and the I/Os. The details of the intra-block workflow is shown in Figure 5.

4 RESULTS AND CONCLUSION

Evaluation setup. We designed an in-house simulator based on our proposed fast-drift model and simulated the raw BER for a 256GB 3D NAND flash. We considered a 40nm process technology and a maximum

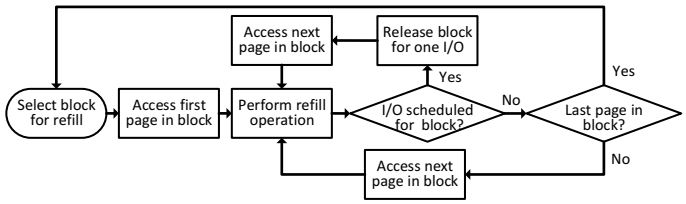


Figure 5: iRefill workflow.

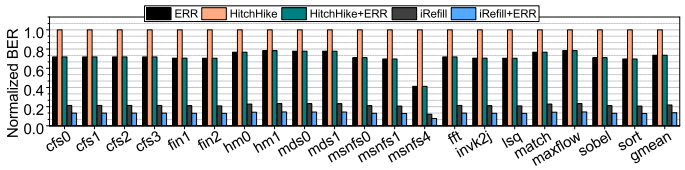


Figure 6: Improvement in normalized bit error rate (BER).

operating temperature of 70°C. We calculated the BER for different configurations running a wide range of real-life workload traces [8]. Using the BER values, we then calculated corresponding ECC latency and energy overheads for a 2.0 bit LDPC scheme [5].

BER improvement. As shown in Figure 6, ERR attains an average BER improvement of 26% over the `Baseline`. Also, `HitchHike` and `HitchHike+ERR` do not show additional BER reduction, since the hitch-hike scheme is not designed to reduce error, but to correct more of them. `iRefill` attains a significant improvement of 78% over the `Baseline`, on average. However, `iRefill+ERR` demonstrates the optimum reliability rating with an average BER improvement of 87%. The combined impact of reducing fast-drift through iRefill, and correcting more errors with ERR, allows `iRefill+ERR` to achieve such excellent reliability.

ECC latency and power reduction. While a robust ECC scheme is required for reliable operation, it also adds significant overhead to the NAND flash. Since the ECC overhead is proportional to the number of errors experienced by the system, reducing BER can significantly lower the ECC latency and power consumption. For example, with the lowest number of error bits to correct among all the configurations, `iRefill+ ERR` produces a 13X latency improvement over the `Baseline`, on average. At the same time, the combined effort of ERR and iRefill reduces the 3D NAND's average ECC energy consumption by 10X.

In conclusion, one can observe that, with the conventional design, emerging 3D NAND flash devices can suffer critical reliability and performance issues. While individually our proposed ERR, Hitch-hike, and iRefill schemes can reduce the problem, combining them can allow 3D NAND flash to meet specific reliability and performance goals.

ORIGINAL PUBLICATION

`M. M. Shihab, J. Zhang, M. Jung, and M. Kandemir. 2018. ReveNAND: A Fast-DriftAware Resilient 3D NAND Flash Design. ACM TACO, 15-2, 17.` https://mustafashihab.com/revenand.pdf

REFERENCES

[1] Chih-Ping Chen et al. 2010. Study of fast initial charge loss and it's impact on the programmed states Vt distribution of charge-trapping NAND Flash. In *IEEE IEDM*.
[2] Bongsik Choi et al. 2016. Comprehensive evaluation of early retention characteristics in tube-type 3-D NAND Flash memory. In *IEEE VLSI Technology*.
[3] Laura M Grupp, John D Davis, and Steven Swanson. 2012. The bleak future of NAND flash memory. In *USENIX FAST*.
[4] Jaehoon Jang et al. 2009. Vertical cell array using TCAT technology for ultra high density NAND flash memory. In *IEEE VLSI Technology*.
[5] Jonghong Kim et al. 2012. Low-energy error correction of NAND Flash memory through soft-decision decoding. *EURASIP JASP* 1 (2012), 195.
[6] Xinkai Li et al. 2014. Investigation of charge loss mechanisms in 3D TANOS cylindrical junction-less charge trapping memory. In *IEEE ICSICT*.
[7] HT Lue et al. 2005. Novel soft erase and re-fill methods for a P+ poly gate nitride-trapping NVM device with excellent endurance and retention properties. In *IRPS*.
[8] Dushyanth Narayanan et al. 2009. Migrating server storage to SSDs: analysis of tradeoffs. In *ACM ECCS*.
[9] Ki-Tae Park et al. 2014. Three-dimensional 128Gb MLC vertical NAND flash-memory with 24-WL stacked layers and 50MB/s high-speed programming. In *IEEE ISSCC*.
[10] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT Press Cambridge.
[11] SungJin Whang et al. 2010. Novel 3-dimensional Dual Control-gate with Surrounding Floating-gate (DC-SF) NAND flash cell for 1Tb file storage application. In *IEEE IEDM*.
[12] Doe Hyun Yoon and Mattan Erez. 2010. Virtualized and flexible ECC for main memory. In *ACM SIGARCH Computer Architecture News*, Vol. 38. 397–408.