SARO: A State-Aware Reliability Optimization Technique for High Density NAND Flash Memory

Myungsuk Kim
Department of Computer Science and Engineering
Seoul National University
morssola75@davinci.snu.ac.kr

Youngsun Song
Department of Computer Science and Engineering
Seoul National University
ysunsong@davinci.snu.ac.kr

Myoungsoo Jung
School of Integrated Technology
Yonsei University
mj@camelab.org

Jihong Kim
Department of Computer Science and Engineering
Seoul National University
jihong@davinci.snu.ac.kr

ABSTRACT

Recent advances in flash technologies, such as scaling and multi-leveling schemes, have been successful to make flash denser and secure more storage spaces per die. Unfortunately, these technology advances significantly degrade flash's reliability due to a smaller cell geometry and a finer-grained cell state control. In this paper, we propose a state-aware reliability optimization technique (SARO), new flash optimization that improves the flash reliability under diverse scaling and multi-leveling schemes. To this end, we first reveal that reliability-related flash errors are highly *skewed* among flash cell states, which was not captured by prior studies. The proposed SARO exploits then the different per-state error behavior in flash cell states by selecting the most error-prone flash states (for each error type) and by forming narrow threshold voltage distributions (for the selected states only). Furthermore, SARO is applied only when the program time gets shorter because of flash cell aging, thereby keeping the program latency unchanged. Our experimental results with real MLC and TLC flash devices show that SARO can reduce a significant number of flash bit errors, which can in turn reduce the read latency by 40%, on average.

ACM Reference Format:
Myungsuk Kim, Youngsun Song, Myoungsoo Jung, and Jihong Kim. 2018. SARO: A State-Aware Reliability Optimization Technique for High Density NAND Flash Memory. In *GLSVLSI'18: 2018 Great Lakes Symposium on VLSI, May 23–25, 2018, Chicago, IL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3194554.3194591

1 INTRODUCTION

Many recent advances and innovations in flash technologies (such as 3D flash, 10-nm node process, and TLC cells) enabled dramatic increases in the flash memory capacity. However, the fundamental relationship between the capacity and reliability of flash memory remains unchanged: *the higher the flash chip capacity, the lower the flash reliability* [1]. As the feature size of technologies shrinks, neighboring flash cells are getting closer and located in a smaller space geometry. Unfortunately, a narrow spacing between adjacent cells tend to be more vulnerable to noise effects such as cell-to-cell interference, thus lowering the flash reliability [1]. In addition, the m-bit multi-leveling technique (e.g., 3-bit triple-level cell (TLC)) significantly lowers the flash reliability because it increases the storage capacity by distinguishing the same threshold voltage (V_{th}) distribution of a cell transistor with 2^m different states. Since a higher m allows a smaller noise margin (between the adjacent cells), the m-bit multi-leveling technique degrades the flash reliability, compared to the single-level cell (SLC) flash device.

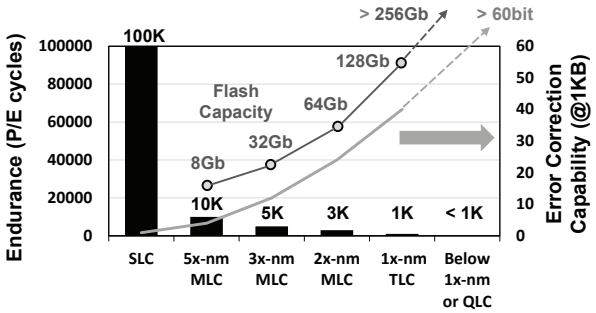

Figure 1: Trends in the flash capacity and flash reliability over advances in flash technologies.

Figure 1 shows how the flash reliability has deteriorated by process scaling and m-bit multi-leveling techniques, which have been the driving force behind the recent growth in the storage capacity in flash. For example, while flash's storage capacity was increased from 8G bits to 128G bits, the flash endurance has been worsened by 10 times. To handle the poor reliability introduced by such advanced techniques, a flash controller should employ 40-bit error correction codes (ECC) rather than 4-bit ECC at the system-level.

To secure a better reliability in flash, a straightforward approach is to keep V_{th} distributions of adjacent program states as distant as possible so that the interference between neighboring cells can be minimized. One simple technique is to reduce the V_{th} width of each program state. Although narrowing V_{th} distributions for program states makes program states *error-resistant*, it slows down the program speed because a program step voltage (V_{ispp}) should be reduced for forming a narrow V_{th} distribution. Since V_{ispp} is inversely proportional to the program latency, it is inevitable to sacrifice the program latency if the reliability is improved in a way that reduces V_{ispp}. Furthermore, many emerging applications (such as real-time big-data analytics, autonomous cars and virtual-reality applications) require both high capacity and high performance from

[1]Over 2D flash, 3D flash is more reliable with a higher capacity. However, within 3D flash, this relationship still holds.

GLSVLSI'18, May 23–25, 2018, Chicago, IL, USA

ACM ISBN 978-1-4503-5724-1/18/05…$15.00
https://doi.org/10.1145/3194554.3194591

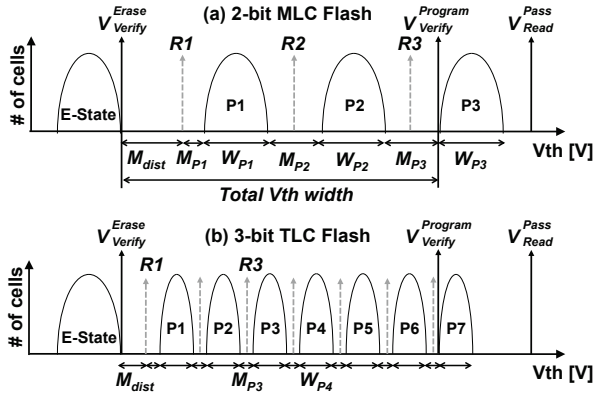

Figure 2: V_{th} distributions of 2^m-state flash memory.

flash-based storage systems, thus making it even less desirable to improve the flash reliability by slowing down the program latency.

In this paper, we propose a new NAND flash reliability optimization technique that addresses the reliability issue of high density flash memories without sacrificing their program latency. Our proposed technique, called as state-aware reliability optimization (SARO), is motivated by our key finding that reliability-related flash errors are highly skewed among program states. That is, not all flash reliability errors occur equally likely among different program states. To be precise on the per-state error characteristics, we performed comprehensive evaluations using 2x-nm TLC flash chips and validated that a few program states are responsible for most reliability errors. SARO narrows V_{th} distributions of these error-prone states only, thus significantly reducing the overheads imposed by applying a fine-grained V_{ispp}. Our SARO also exploits a well-known flash phenomenon that the program latency is decreased as flash cells get older because of traps in the damaged tunnel oxide [2]. Specifically, we employ a fine-grained V_{ispp} only when the program latency gets shorter, which can in turn make SARO achieve high flash reliability while keeping the program latency unchanged.

For the validation of SARO, we built per-state error models using 2x-nm TLC flash chips. Based on the per-state error models, we devise a simple but effective method of selectively modifying V_{ispp} values under given flash cell status (such as P/E cycles). Our experimental results show that SARO can improve the reliability of MLC and TLC flash devices by 100% and 50%, respectively. As a direct result of the improved reliability, SARO shortens the read latency by 40%, on average, by reducing the number of read retry operations.

The rest of this paper is organized as follows. Before SARO is presented, we explain the key trade-off relationship between the reliability and program latency in flash devices using real MLC and TLC flash chips. The key schemes of SARO are presented in Section 3. Section 4 describes the SARO implementation details. Experimental results follow in Section 5, and related work is summarized in Section 6. Section 7 concludes with a summary and future work.

2 BACKGROUND

2.1 Reliability vs. V_{th} Design Parameters

For writes, flash changes target cell's V_{th} states based on the content information (per bit). In the case of reads, flash reconstructs the content by sensing the V_{th} state of target cells. Figure 2 illustrates V_{th} distributions for 2^m-state flash devices, which stores m bits within a cell by using 2^m distinct V_{th} states (i.e., m is 2 and

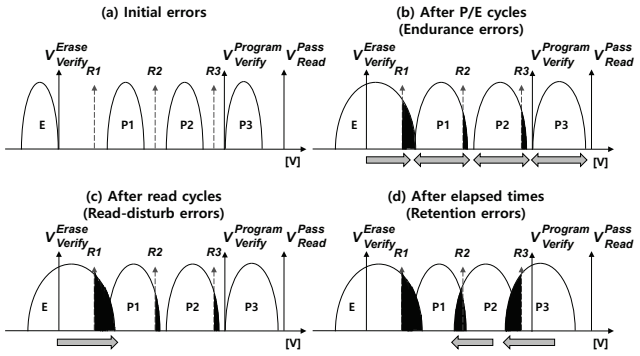

Figure 3: Change patterns in V_{th} distributions depending on different error types.

3 for MLC and TLC, respectively). Since the primary V_{th} design parameters shown in Figure 2 (i.e., W_{Pi}'s, M_{Pi}'s, M_{dist}, V_{Verify}^{Erase}, and $V_{Verify}^{Program}$) directly affect both the flash reliability and flash performance, how to set these parameters is a critical decision for NAND flash manufacturers. Once the V_{th} design parameters are fixed during the flash design time, the flash reliability largely depends on how V_{th} distributions are disturbed by various causes. Bit errors in flash memory occur when the V_{th} distribution of one state is shifted to neighboring state's V_{th} regions so that two overlapping V_{th} distributions cannot be distinguished as two different states.

In general, we can categorize bit errors in flash into four groups as shown in Figure 3. When a page is written, a small number of bit errors are inevitable as shown in Figure 3(a). These initial errors are caused by process variations and inherent noise effects (such as cell-to-cell interference, back pattern dependency, and random telegraph noise) [3]. More serious bit errors, which mandate to employ a strong error-correction scheme (such as LDPC codes) in a flash controller, are related to defects in the tunnel oxide layer of flash cells [4]. As a flash device experiences P/E cycles, the width of V_{th} distributions tend to get wider because of traps in tunnel oxide layer which are caused by applying a high program/erase voltage frequently. The traps in the tunnel oxide layer generated by P/E cycles also exacerbate the program disturbance. When flash cells are programmed, neighboring cells belonging to the E-state are softly programmed, so that their V_{th} move to the right. As shown in Figure 3(b), these errors are called endurance errors. The repeated read operations also shift V_{th} of E-state cells to the right. When flash cells are read, V_{Read}^{Pass} (6~7V) is applied to all the wordlines (WLs) in the same block except for the target wordline [2]. When pages in the same block are frequently read, E-state cells in the block are softly programmed, thus shifting their V_{th}s to the right. These errors, as shown in Figure 3(c), are called read-disturb errors. If flash cells are left for a long time after a program operation, a charge loss occurs by the stress-induced leakage current (SILC) through traps in the tunnel oxide layer. These errors, as shown in Figure 3(d), are called retention errors. The charge loss shifts the V_{th} of program states to the left. Furthermore, higher program states experience a greater amount of the charge loss, and hence show a larger V_{th} shift compared to lower program states [5].

When V_{th} distributions are sufficiently moved, V_{th} distributions may overlap with read reference voltages (i.e., R1, R2 and R3 in Figure 3), thus producing flash bit errors. For example, the shaded regions in Figure 3 indicate such overlapped V_{th} distributions. If

[2]The memory cells are packed to form a matrix structure. As shown in Figure 9, cells share a wordline (WL) horizontally and a bitline (BL) vertically.

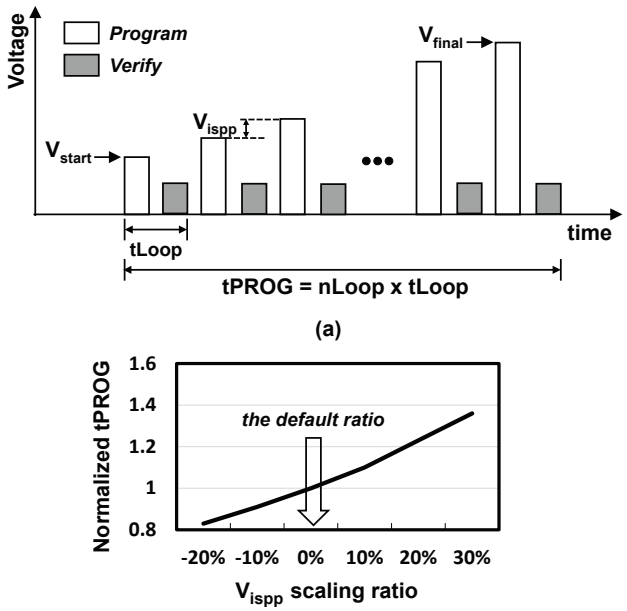

Figure 4: An overview of the ISPP scheme: (a) Key parameters for *tPROG* and (b) Normalized *tPROG* over varying V_{ispp} scaling ratios

there is an insufficient V_{th} margin between the states, flash bit errors sharply increase as V_{th} distributions are disturbed. When the number of flash bit errors reaches over a pre-defined upper limit, which is generally determined by the error correction capability of an ECC engine in a flash controller, the flash page cannot be correctly read, resulting in a read-failure status. Thus, to guarantee the reliability of a flash device, sufficient M_{dist} and M_{Pi} should be secured considering the maximum V_{th} changes after the maximum number of P/E cycles, read cycles, and retention times.

2.2 Reliability vs. Program Latency

Typically, a flash device employs an *incremental step pulse programming* (ISPP) scheme [6] to control the width W_{Pi} of program states. As shown in Figure 4(a), the ISPP scheme gradually increases the program voltage by V_{ispp} at a time until all their cells in a page are completely programmed. For each program loop that takes *tLOOP*, once cells are verified to have been sufficiently programmed by a verify operation, those cells are excluded from the next program loop. Since the width W_{Pi} of the i-th program state is proportional to V_{ispp}, a simple solution to ensure sufficient M_{dist} and M_{pi} is to narrow W_{Pi} by reducing V_{ispp} during a program operation. Even though the V_{ispp} reduction can improve the flash reliability, it increases the total number of *nLoop* of program loops because a smaller V_{ispp} value needs to repeat the program loop more to reach V_{final} for the maximum program state. Therefore, the program latency is inversely proportional to V_{ispp}. Figure 4(b) shows that the program latency directly increases as V_{ispp} is scaled down.

Figure 5 shows the results of reliability test over various V_{ispp} scaling ratios using real 2x-nm node MLC and TLC chips [3]. (For a detailed description on the evaluation settings, refer to Section 5.) As expected, both the endurance and retention capability were significantly improved with a smaller V_{ispp} value. In 2x-nm MLC flash chips, the maximum number of P/E cycles is increased from less than 7,000 to more than 10,000 when V_{ispp} is reduced by 30% (i.e., mode*30*). The data retention capability is also increased under mode*30* from about 2 years to 5 years [4].

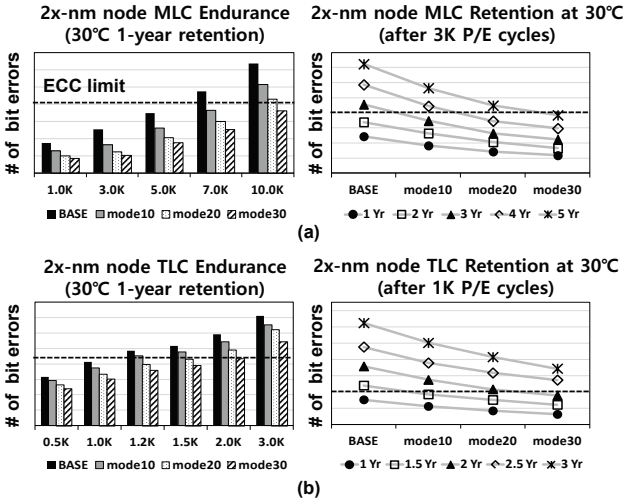

Figure 5: Reliability of flash devices over various V_{ispp} scaling ratios: (a) 2x-nm MLC flash and (b) 2x-nm TLC flash

As shown in Figure 4 and 5, there is a trade-off relationship between the flash reliability and the flash program latency when V_{ispp} varies. Our main goal of the proposed SARO technique is to investigate if we can achieve the same flash reliability improvement from a smaller V_{ispp} value without sacrificing the program latency.

3 SARO: BASIC IDEA

3.1 State-Aware Reliability Characterization

Since the program latency is proportional to *nLoop* and *nLoop* is, in turn, inversely proportional to V_{ispp}, reducing V_{ispp} directly increases the program latency and vice versa. In order to mitigate the impact of a smaller V_{ispp} on the program latency, we investigated whether flash errors occur in a *state-oblivious fashion* or in a *state-aware fashion* using 2x-nm TLC flash chips. Our intention was that if flash bit errors occurred differently over different program states, we can apply a smaller V_{ispp} value only to those states with higher probability of bit errors while not reducing V_{ispp} for the other states. By combining a small V_{ispp} value with a larger one, we can effectively lessen the impact of V_{ispp} on the program latency while, hopefully, achieving the same reliability improvement.

Figure 6 shows a breakdown of per-state flash bit errors on our characterization study with 2x-nm TLC flash. Our results strongly indicate that flash bit errors occur in a *state-aware fashion*. That is, different error types occur in different program states. Most of endurance and read-disturb errors occur between the E and P1 state by the overlapped V_{th} distribution of the E-state cell and R1 read reference voltage level. In Figure 6, about 43% of the total endurance and read-disturb errors occur between E and P1 states. On the other hand, most of retention errors occur in higher states. In Figure 6,

[3]When the V_{ispp} scaling ratio is set to *X*%, V_{ispp} is reduced by *X*% of the nominal V_{ispp}. We call this V_{ispp} setting by mode*X*. BASE represents the baseline technique where V_{ispp} scaling ratio is 0%.

[4]The data retention capability is measured under the retention temperature of 30°C after the maximum P/E cycles (3,000 for MLC flash and 1,000 for TLC flash) and 1,000 read cycles for each block. This evaluation setting is a de facto standard for NAND manufactures.

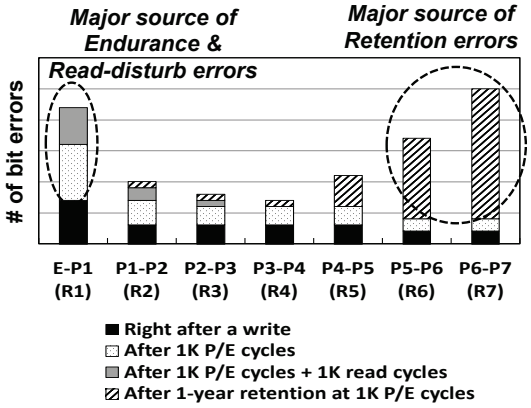

Figure 6: A breakdown of per-state error types on 2x-nm TLC flash.

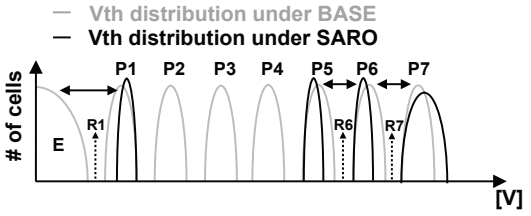

Figure 7: An overview of the state-aware selective programming scheme on a TLC flash device.

retention errors between the P5 and P6 states account for about 31% of the total retention errors while retention errors between the P6 and P7 states are responsible for 50% of the total retention errors.

Based on our characterization results, SARO employs a state-aware program scheme, optimizing each program state based on its own reliability requirement. As shown in Figure 7, we apply the reduced V_{ispp} value to most error-prone states (e.g., P1, P5, and P6 in Figure 7) only so that most bit errors can be avoided with a small increase in the program latency. In addition to the reliability optimization from a finer V_{ispp} control, SARO improves the read latency as well. For example, in Figure 7, as M_{P1}, M_{P6}, and M_{P7} get wider, we adjust R1, R6, and R7 accordingly. This adjustment contributes to reducing the number of read retry operations required. Furthermore, since P7 has no right-side neighbor, we *increase* its V_{ispp} so that P7 program latency can be further reduced.

Our SARO technique can be easily generalized to 2^m-state flash as follows:

- Reduce V_{ispp} and increase the verify and read reference voltage (Rx) of the lowest program state (i.e., P1) to improve the endurance and read disturb error.
- Reduce V_{ispp} of program states from (2^m-2) state to $2^m-(2^{m-2}+1)$ state to improve the retention error.
- Increase V_{ispp} of (2^m-1) (i.e., the highest) program state to reduce the performance degradation.
- Decrease Rx of program states from (2^m-1) state to (2^m-2^{m-2}) state to make the read latency shorter.

3.2 Slack-Aware Adaptive NAND Programming

In order to minimize the performance degradation from reducing V_{ispp}, we devised a slack-aware scheme which accurately estimates the *tPROG* acceleration ratio as flash cells get aged. For an accurate

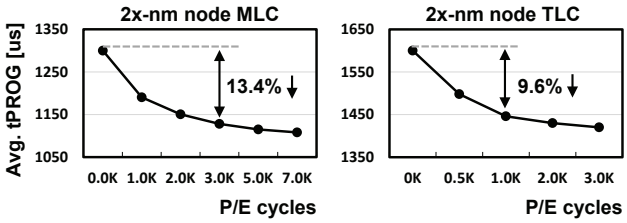

Figure 8: *tPROG* speed changes over varying P/E cycles.

estimate of *tPROG*, we performed a flash characterization study while varying the number of P/E cycles. As shown in Figure 8, the average latency for a page programming constantly decreases as the number of P/E cycles increases. In MLC flash devices, after applying 3,000 P/E cycles, the average program latency decreases by 13.4%, compared to an initial value (1,300 μs). The average program latency of TLC flash devices also decreases by 9.6% after 1,000 P/E cycles. The effect of *tPROG* acceleration over P/E cycles can be explained using traps in the tunnel oxide layer. As the number of P/E cycles increase, traps also increase because a high program/erase voltage damages the tunnel oxide layer. As the tunnel oxide layer gets damaged, its insulation capability gets worse, thus allowing electrons to move easily into flash floating gate. As a consequence, programming the aged pages (e.g., experienced a lot of P/E cycles) requires lesser number of program loops to complete the program operation.

Based on the *tPROG* acceleration model, SARO applies a finer V_{ispp} value only when *tPROG* gets faster. In order not to introduce any latency overhead, SARO tries to match an increase from a finer V_{ispp} value with a decrease in *tPROG*. Although this solution is helpful to improve the flash reliability and easy to implement without any hardware modification, if we apply the proposed slack-aware V_{ispp} adjustment technique only without exploiting per-state error behavior of Section 3.1, its effect on the flash reliability is rather limited. Since the decrease of the program latency (even) at the end of the flash lifetime is about 10% of the default program latency, V_{ispp} can be reduced up to 10% only. For TLC flash, the reliability improvement is marginal, i.e., about 10% only, as shown in Figure 5(b).

4 SARO: IMPLEMENTATION DETAILS

4.1 State-Aware Selective NAND Programming

To support state-aware programming, a flash device should apply different V_{ispp} values to cells in the same WL. Fortunately, since NAND flash memory supports a bit-by-bit selective operation with the self boosting program inhibit (SBPI) scheme [6], state-aware selective programming can be easily implemented by modifying the BL operating conditions without an additional hardware modification.

Figure 9 illustrates how the SBPI scheme supports bit-by-bit selective program operations. When a page WL_k is programmed, its i-th cell within the page is selectively programmed depending on the value of BL_i. If BL_i is set to 1 (i.e., Vcc), the i-th cell is not programmed (i.e., inhibited) because there is an insufficient voltage difference by channel boosting effect (P1 in Figure 9(a)). If BL_i is set to 0 (i.e., 0V), the i-th cell is programmed because there is an enough voltage difference to move the electron into the floating-gate (P6 in Figure 9(b)). Based on a conventional SBPI, when the i-th cell is programmed, the *effective* V_{ispp} for the i-th cell during a program operation can be reduced by forcing a specific voltage into the channel connected to BL. For example, when a program

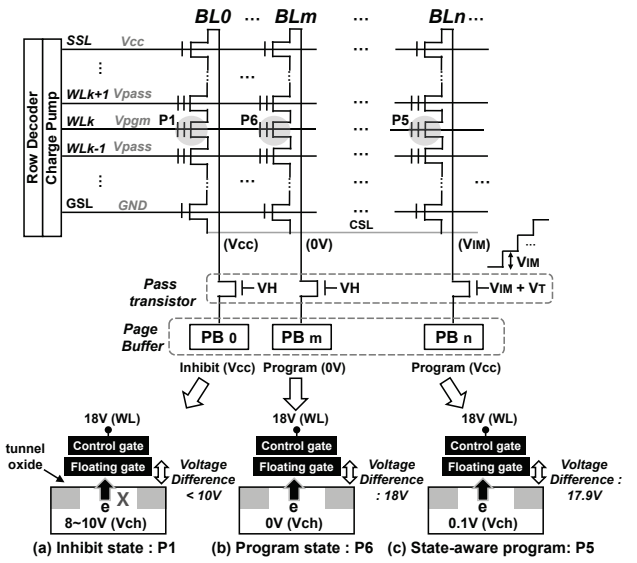

Figure 9: Design Schematics to implement state-aware selective programming using conventional SBPI scheme and page buffer operation.

voltage is increased from 18V to 18.4V by V_{ispp} (= 0.4V), the *effective* V_{ispp} can be reduced to 0.3V by applying 0.1V to the channel. In our example, P1 state is inhibited (Figure 9(a)), P6 state is normally programmed by 0.4V V_{ispp} (Figure 9(b)), and P5 state in BL_n is programmed in a state-aware fashion (Figure 9(c)). At the start of P5 state programming (assuming that the start program voltage of P5 state is 18V), the WL gate is set to 18V by the target row decoder, and the PB_n forces Vcc to the BL_n. Since the gate of the pass transistor connected with BL_n is set to V_{IM}+V_T (i.e., V_T is a threshold voltage of the pass transistor), the voltage which can be transfered to the BL_n becomes V_{IM}. Therefore, the voltage difference between the channel and floating-gate of P5 state is reduced by V_{IM}, which indicates that the effective V_{ispp} is 0.4V−V_{IM}. As the program loops of the target program state proceed, the gate voltage of the pass transistors is increased by V_{IM} to maintain the reduced effective V_{ispp}.

4.2 SaFTL: SARO-aware FTL

In order to take advantage of the SARO technique in flash-based storage systems, we modified an existing page-level FTL to support SARO. (We call this modified FTL `SaFTL`, SARO-aware FTL.) The main goal of `SaFTL` is to manage various operating parameters based on the program latency measured from NAND flash memories. Figure 10 shows an overall organization of `SaFTL`. `SaFTL` monitors the program latency from a flash device. The measured program latency is compared to its initial (non-accelerated) *tPROG*. Depending on the difference between the measured *tPROG* and the initial *tPROG*, the *Mode Selector* module decides a proper *Mode* for a program request. For example, if the measured *tRPOG* is faster than the initial *tPROG* by 3%, the next program request is programmed using *Mode* 3 entry of the *Parameter Table*. The *Parameter Table* contains all the necessary values for V_{th} design parameters for a given *Mode* type.

In order for `SaFTL` to properly work, the initial program latency and *Mode* should be maintained for every programmed page. However, in order to reduce the memory overhead for implementing

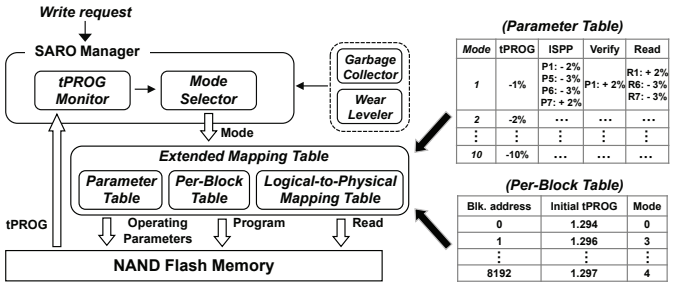

Figure 10: An Organizational overview of `SaFTL`

SARO, a mode selection is applied in a per-block basis. (That is, all the pages in the same block is programmed using the same *Mode*.) Since variations in *tPROG* within the same block is negligible (e.g., less than 3 μs in our evaluations), per-block mode selection is an effective solution to minimize both the memory overhead and the *tPROG* monitoring overhead.

In order to maintain up-to-date status per each block, `SaFTL` uses the *Per-Block Table* which can be implemented with a small memory overhead. For example, a flash device with 8,192 blocks needs extra 22-KB memory for *Per-Block Table*. The initial program latency is measured when the flash device is first programmed, and stored in a pre-defined area. The current program latency is determined by monitoring only one last WL per block.

5 EXPERIMENTAL RESULTS

We conducted experiments using 20 real 2x-nm TLC and MLC flash chips, respectively, to evaluate the effectiveness of the proposed optimization technique. In order to minimize the potential distortion of the evaluation results from process variations, we evenly selected 30 test blocks from each chip at different physical block locations, and tested all the pages in each selected test block. As a result, a total of 115,200 pages (on TLC flash chips) and 76,800 pages (on MLC flash chips) were used to obtain statistically significant experimental results. Using an in-house custom test board (equipped with a NAND controller and a thermal controller), we conducted comprehensive reliability and performance evaluations varying key variables such as P/E cycles, read cycles, and retention times. For example, in MLC flash chips, the endurance test was performed with increasing the P/E cycles from 0 (an initial condition) to 10,000. Similarly, the retention capability was evaluated by changing the elapsed time (since the program operation) from 1 year to 5 years.

Figure 11 (a) and (b) show how the flash endurance and retention capability change under four different operation modes, BASE, mode*10*, mode*30*, and SARO. SARO improves endurance and retention capability by 100% in MLC flash chips and 50% in TLC flash chips over BASE, respectively. To better understand why SARO outperforms BASE, we measured flash bit errors in each pages. The number of flash bit errors in SARO is reduced by 56.1% in MLC flash chips and 25.2% in TLC flash chips over BASE because SARO increases the V_{th} margin between an error-prone adjacent states, thus improving the error tolerance of a flash device. As expected, mode*30* shows the best reliability improvement, but it also degrades the program latency by about 20% as shown in Figure 11(c). SARO, on the other hand, barely affects the program latency. For TLC flash chips, SARO increases the program latency by 1.8% which is negligible overhead.

In order to understand the impact of the improved reliability on the read latency, we evaluated how the number of read retry operations changes under SARO. The read retry operation is initiated

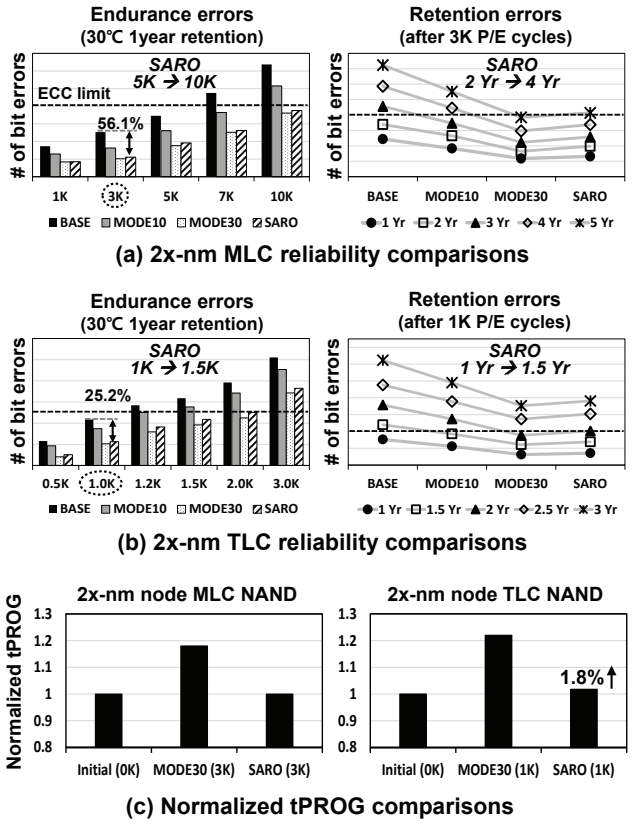

Figure 11: Comparisons of reliability and performance metrics of four different operation modes.

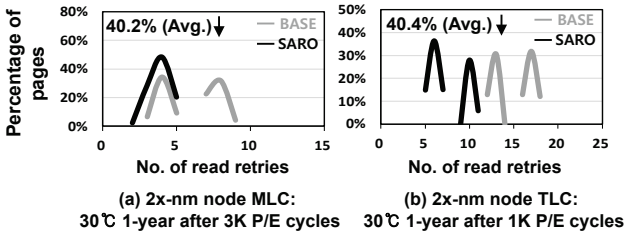

Figure 12: Changes in the number of read retry operations under SARO **over the baseline technique.**

when uncorrectable errors are detected, and repeated until errors can be corrected by the ECC scheme. Since the read latency of a flash device is directly proportional to the number of repeated read retry operations, the read latency is significantly improved when the number of read retry operations is reduced. Figure 12 shows the number of read retry operations under BASE and SARO in MLC and TLC flash chips. SARO reduces the average number of read retry operations by 40% on average over BASE.

6 RELATED WORK

There have been several studies that attempt to improve the reliability of flash devices. Pan *et al.* [7] proposed a technique which can reduce V_{ispp} with P/E cycles to optimize the reliability and performance of NAND flash memory. However, this technique changes V_{ispp} of all program states without considering different error patterns among different program states. Therefore, this technique is rather limited to apply for high-density flash devices, such as TLC flash devices. Park *et al.* [8] also proposed a technique which can change program operating parameters according to P/E cycles. However, their technique is different from SARO in that they did not explore per-state error characteristics. In addition, their technique targets SLC flash devices only so that it cannot be applied for high-density flash devices. The *DPES* technique proposed by Jeong *et al.* [9] is similar to our SARO in that it dynamically changes erase and program parameters of a flash device. However, unlike SARO, it improves the flash endurance only by reducing a stress to flash cells. SARO is different from this technique in that SARO improves the flash reliability in a comprehensive fashion by considering all the different error types including the endurance errors.

7 CONCLUSIONS

We have presented a new flash reliability optimization technique, SARO, that solves the reliability problem of high-density NAND flash memory without sacrificing the program latency. Our proposed technique is based on the state-aware selective programming scheme, which enables each program state to be programmed by a different V_{ispp} value. Since SARO improves the error tolerance of a flash device by selectively reducing V_{ispp} for the most error-prone NAND states only, it is an effective solution to maximize the reliability improvement while minimizing the performance degradation. In order to keep the performance unchanged, SARO parameters are fine-tunned exploiting the flash characteristics that the program latency decreases with cell aging. Our evaluation results with real flash chips show that SARO is effective in improving the flash reliability thanks to the significant decrease in NAND bit errors. It also improves the read latency by 40% on average due to the reduction of read retry operations.

The current version of SARO can be extended in several directions. For example, since a finer V_{ispp} value is more effective for higher flash reliability, we plan to better exploit various system hints (e.g., latency-insensitive program operations) to more aggressively reduce V_{ispp} without affecting the program latency.

ACKNOWLEDGMENT

This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1701-11.

REFERENCES

[1] Y. Park et al. 2014. Scaling and Reliability of NAND Flash Devices. In *Proc. IEEE Int. Reliability Physics Symposium.*
[2] Y. Woo et al. 2013. Diversifying Wear Index for MLC NAND Flash Memory to Extend the Lifetime of SSDs. In *Proc. Int. Conf. on Embedded Software.*
[3] R. Micheloni et al. 2010. Inside NAND Flash Memories. Springer.
[4] N. Mielke et al. 2017. Reliability of Solid-State Drives Based on NAND Flash Memory. In *Proc. IEEE*, Vol. 105(9). 1725–1750.
[5] Y. Cai et al. 2015. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *IEEE Int. Symposium on High Performance Computer Architecture.*
[6] K. Suh et al. 1995. A 3.3V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme. In *IEEE Tran. on Consumer Electronics*, Vol. 30(11). 1149–1156.
[7] Y. Pan et al. 2011. Exploiting Memory Device Wear-Out Dynamics to Improve NAND Flash Memory System Performance. In *Proc. USENIX Conf. on File and Storage Technologies.*
[8] S. Park et al. 2012. Adaptive Program Verify Scheme for Improving NAND Flash Memory Performance and Lifespan. In *IEEE Asian Solid-State Circuits Conf.*
[9] J. Jeong et al. 2014. Lifetime Improvement of NAND Flash-based Storage Systems Using Dynamic Program and Erase Scaling. In *Proc. USENIX Conf. on File and Storage Technologies.*