Rethinking WOM Codes to Enhance the Lifetime in New SSD Generations

Shehbaz Jaffer Kaveh Mahdaviani Bianca Schroeder
University of Toronto

Abstract

New generations of Solid State Drives (SSDs) offer increased storage density with higher bits per cell, but an order of magnitude lower Program and Erase (P/E) cycles. This decreases the number of times one can rewrite on the SSD, and hence, the overall lifetime of the drive. One way of improving drive lifetime is by applying Write-Once Memory (WOM) codes which can rewrite on top of pre-existing data without erasing previous data. This increases the total logical data that can be written on the physical medium before an erase operation is required. Traditional WOM codes are not scalable and only offer up to 50% increase in total writable logical data between any two erase operations. In this paper we present a novel, simple and highly efficient family of WOM codes. Although our design is generic and could be applied to any N-Level cell drive, we focus on QLC drives to demonstrate and evaluate the proposed scheme and show that it can increase the total logical writable data before an erase in a range of 50-375% the physical medium capacity with varying storage overheads. Next, we argue that it is possible to further increase the total logical writable data between two erase operations by up to 500% with the help of a carefully chosen internal error-correcting code (ECC) already present in SSDs.

1 Introduction

Flash-based Solid State Drives (SSDs) offer a faster alternative to Hard Disk drives (HDDs), but have a major limitation: unlike HDDs where previously written data is over-writable, a flash cell needs to be erased before it can be programmed, and each erase operation causes wear-out that reduces a cell's lifetime. Older generations of flash were based on single-level cells (SLC), which store only a single bit in a cell and can typically tolerate several thousand program erase cycles before wearing out. However, to keep up with the increasing demand for storage capacity, more bits need to be stored in a cell. Recent work [25] shows that with each additional bit stored in 1 SSD cell, the number of erase cycles that the SSD can endure reduces by an order of magnitude. Figure 1 illustrates the problem based on recent projections [2]. Flash based on Multi-level Cells (MLC) and Triple Level Cells (TLC), which are are common nowadays, can tolerate a significantly smaller number of P/E cycles. Even more worrisome is a look into the future with QLC and PLC drives, which might see P/E cycle limits drop to tens or a few hundred. To make high-density SSD drives such as QLC drives usable beyond archival applications, it is paramount to reduce the number of times the storage media is erased.

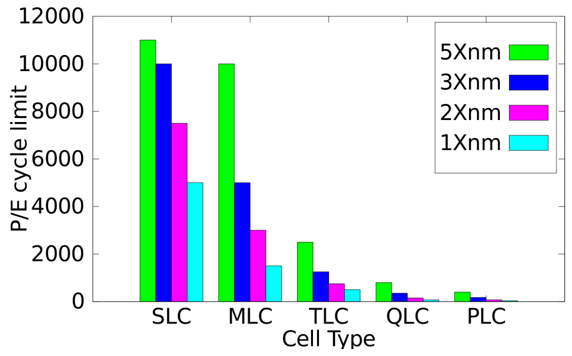

Figure 1: *P/E cycles decrease with increased storage density [2]. Smaller sized cells can endure lower number of P/E cycles within the same SSD type.*

One way to reduce the number of times we erase a flash block before storing new data is to use WOM codes. WOM codes can be overwritten on top of each other without erasing the previously written code. In this paper **our position** is that traditional WOM codes, as they have been used in previous work to extend flash lifetime, are not efficient for the specific characteristics of flash with multiple levels and that great improvements are possible with a better code design. More precisely, we observe that traditional WOM codes impose a bit-level constraint on each code, where each bit in a codeword can only be set from 0 to 1 while it is being overwritten. This constraint is not only a sub-optimal fit for flash with multiple levels, it also limits the number of code-words that can be overwritten as media becomes denser and stores more bits per cell and is therefore not scalable. This significantly reduces the number of overwrites that can be done before an erase operation is required. Further, several other codes like Polar WOM Codes [33] involve multiple iterations for encoding and decoding data which is complicated and inefficient.

In this paper, we propose a *highly efficient* family of WOM codes for QLC flash that can be implemented using simple *table lookups*. Our scheme is generic and can be extended to TLC, PLC or any N-Level Cell SSD drive. First, we present a family of WOM codes that model QLC cells as monotonically increasing voltage values instead of a collection of bits that can only be set from 0 to 1. This creates a more efficient WOM coding scheme by utilizing all intermediate voltage levels, and helps to improve the additional logical writable capacity of the disk before erase, compared to uncoded writes, from 50% to 375%. Second, we show that by using a carefully chosen internal ECC embedded within flash, typically used to correct flash raw bit errors, we may be able to extend the amount of logical writable capacity of drive before erase to 500% the physical disk capacity.

2 Limitations of traditional WOM codes

data	Gen1	Gen2
00	000	111
01	100	011
10	010	101
11	001	110

Table 1: *WOM(2,3)*

data	Gen1	Gen2	Gen3
00	0 000	1 000	1 111
01	0 100	1 100	1 011
10	0 010	1 010	1 101
11	0 001	1 001	1 110

Table 2: *WOM(2,4)*

Historically, Write Once Memory (WOM) Codes were motivated by media, such as punch cards and optical disk, where data is written at the granularity of bits, and once written a bit can only be changed in one direction, e.g. from 0 to 1. Seminal work by Rivest and Shamir [22] presents a way to accommodate multiple writes on such media by encoding x bits of (*data bit sequences*) into a *code-word* of y bits such that a certain number of overwrites in place are possible. This is referred to as a WOM(x,y) code.

Table 1 shows a simple example of a WOM(2,3) code, which can guarantee at least two writes in place. The first write uses the mapping from data bit sequences to code-word given in the column labelled Generation 1, and the second write uses the mapping in the column labelled Generation 2. For example, to consecutively write the two data bit sequences 01 and 10 in place, one would write the physical words 100 (Generation 1) and 101 (Generation 2). Note that the second write only requires valid transitions of bits from 0 to 1.

It's easy to verify in Table 1 that any data bit sequences encoded in Generation 1 can be overwritten by any other data bit sequences encoded in Generation 2, with only valid bit transitions from 0 to 1. That means we can always guarantee at least one overwrite in place. The actual number of times we can write in place can be larger than the number of generations if the data bit sequence contains consecutive writes of the same data bit sequence, which don't require changes to the physical medium. We call each time data is transferred to the media for writing as one *Write Cycle* (WC). For example, writing the 4 data bit sequences 01, 01, 10, 10 requires only two writes to the medium (the same as in the previous example: 100 and 101), but count as 4 Write Cycles.

The assumption of being able to change a bit in only one direction (from 0 to 1) matches the characteristics of a (single-level) flash cell and inspired prior work [18, 28, 30, 33] to use WOM codes to increase the lifetime of flash. However, these prior applications of WOM codes have several limitations:

Modelling cells as group of bits

WOM codes, as described before in the context of flash, work at bit-level and assume bits can only be changed from 0 to 1. We call such codes *bit based codes*. Prior work has explored non-binary WOM codes [11] but not in the context of QLC flash. The bit-based model is not a good fit for modern flash, which stores multiple bits in a cell. For example, a QLC cell distinguishes 16 different voltage levels, where each voltage level represents a 4 bit binary code. Any transition between different states of a cell is governed by the current voltage level: reducing the voltage level is not possible without first erasing, but increasing is possible. It is not dependent on the individual bit values of the 4 bit value stored in a cell (represented by the current voltage level).

Scalability

The *bit based model* creates unnecessary constraints that limit the usability for QLC flash. In order to show these restrictions, we introduce a naive extension of WOM(2, 3) to a WOM(2, 4) code which encodes any 2 bits of data into 4 *coded bits* to be written in a cell for a QLC drive where all the bit-level constraints are still satisfied. To this end we add a 0 / 1 to all code-words in Gen 1 of WOM(2, 3) (Table 1 column 2) to create Gen 1 / Gen 2 for WOM(2, 4). Next, we add a 1 to all code-words in Gen 2 of WOM(2, 3) to create Gen 3 in WOM(2, 4) (Table 2 bits colored green). Non-Binary WOM codes have been well studied and optimized for various metrics [4, 5, 7, 9–11, 13, 15, 24, 29] but we limit extending to a code-word scheme with 4 bits per code-word and 4 code-words per generation each representing a unique data bit sequence to model a QLC cell. Although WOM(2,4) enables an additional generation before erasure, only 12 out of 16 possible four-bit code-words that could be written in a QLC drive's cell are used in this scheme as we try to maintain the bit-level constraint. This limits the total number of potential generations to 3 instead of 4 generations. The artificial reduction in code-word usability due to bit constraints will only worsen with denser media.

3 Solution

SSD cells that store multiple bits have different characteristics than how WOM codes in the past have modeled them. In [18] Margaglia et. al. examined possible reprogramming operations between different intermediate states in MLC SSDs. Their observation confirms that some transitions are possible which are not compatible with the bit-level constraints.

Moreover, there are cases where the bit-level constraints assume the transition is possible but in practice that transition is not possible due to hardware constraints. In order to design WOM codes that are capable of utilizing the full potential of QLC drives, we consider a different model for WOM code constraints. Although we evaluate our approach on the most recent and dense drive commercially available, our approach is generic and can be extended to any N-Level Cell drive, and also matches better with the actual underlying constraints of the flash cell hardware to the best of our knowledge.

Voltage Based QLC WOM Code

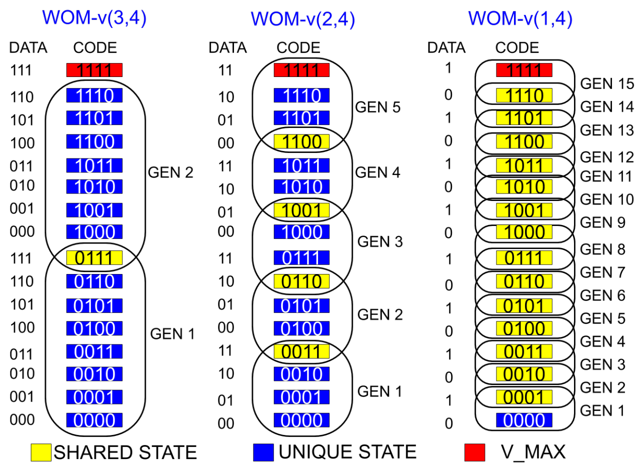

Figure 2: *Voltage based codes for encoding 3 2 and 1 data bit(s) into 4 coded bits respectively. Each oval represents a generation. Each generation contains a set of code-words mapped to a set of all possible data bit sequence for the code.*

In this section we change the model by imposing the invariant constraints for WOM coding in terms of voltage levels stored in flash cells rather than encoded bits. In this model the only invariant constraint that should be satisfied in WOM code reprogram operations is that the voltage level in each flash cell can only be increased or kept unchanged before an erasure. We refer to WOM codes designed based on this constraint model as *voltage based WOM codes*, and denote them by WOM-v(k,n) when the coding scheme maps any k data bits to one of 2^n voltage levels stored in the flash cell.

In a QLC drive, each voltage level could be considered as a sequence of 4 coded bits. In Figure 2, we present 3 instances of a family of WOM-v codes, WOM-v(3,4), WOM-v(2, 4) and WOM-v(1,4). In summary, a WOM-v(k,n) code is a unique mapping between any sequence of k data bits and a set of voltage levels (or equivalently their coded bit sequence representations), referred to as *code-words*. A set of consecutive *code-words* that cover all possible *data bit sequence* make one *generation*. Each coding scheme can have GEN_MAX number of generations, which are 2,5 and 15 in WOM-v(3, 4),WOM-v(2, 4) and WOM-v(1, 4) respectively. All coding schemes have a maximum voltage V_MAX (shown in red) after which the cell needs to be erased.

Consider WOM-v(2, 4) code (Figure 2 center) where 2 data bits are encoded into 4 code bits. Therefore, we have four possible data bit sequences, indicated by 00, 01, 10 and 11. Each generation has 4 *code-words* mapped to a unique *data bit sequence* within a generation. With *voltage based coding*, the following three optimizations help us accommodate 5 generations for WOM-v(2,4) efficiently:

Code-word Sharing

One feature used in the design of this WOM-v code family is that each pair of consecutive generations are *sharing* one common code-word. A *voltage based model* has a linear structure where we are able to overlap *code-words* between successive generations. This compresses all generations leaving space for additional generations. For example, in WOM-v(2, 4) code, without sharing code-words we would only be able to accommodate 4 Generations consisting of voltage levels 0-3, 4-7, 8-11 and 12-15 respectively (not shown). With shared *code-words* between generations, we now can have 5 generations from 0-3, 3-6, 6-9, 9-12, 12-15, (Fig 2) with shared *code-words* 3,6,9 and 12 between consecutive generations (shown in yellow). The data bit sequence corresponding to each shared *code-word* is the same in both generations. Example. *code-word* 0011 maps to data bit sequence 11 in both Gen 1 and Gen 2. Note that having larger GEN_MAX translates into more logical data write cycles before erasure. The gain of code-word sharing are much higher in WOM-v(1, 4), where GEN_MAX increases from 8 disjoint generations, to 15 generations with overlapping states, Fig 2).

Same Generation Transition

Another feature we suggest in the design of this WOM-v code family is to perform reprogram operations by transition to a higher voltage level within the same generation whenever possible. In other words, in each write cycle, we suggest to simply increase the voltage level in each cell to the lowest voltage level corresponding to the data bit sequence we need to write in that cell which is not below the current voltage level in that cell. For example, consider WOM-v(2,4), if data bit sequences 01, 01, 10 and 00 are written in 4 successive write cycles, we would encode and change the voltage level to 0001, 0001, 0010 and 0100 for the write cycles one through four respectively. The second write does not change the first code-word as the corresponding data bit sequences are the same. While writing the third code-word on 0001, instead of transitioning to 0110 in the next higher generation, we increase the voltage level from 0001 to 0010. Finally, the fourth write cycle involves writing data bits 00 so we overwrite 0010 to 0100. Note that without self generation transition, the *code-word* would have the following order 0001, 0001, 0110 and 1000.

As demonstrated in this example, with the same generation transitions we could potentially achieve a significant saving in voltage level increase during write cycles, and therefore, enables us to perform more write cycles before we need an erasure. However, this comes with a drawback, which is addressed in the next subsection. The drawback here is that the voltage level stored in different cells in a page would end up being in different generations after a few write cycles. That is due to the fact that the pattern of voltage level increment in each cell depends on the pattern of data bits written in successive write cycles. This albeit does not affect reading the stored data, since the decoding mapping from voltage levels to data bits is unique.

Reusing underutilized Cells

In this section we present another optimization that further increases the number of writes before an erase becomes unavoidable. Our idea is based on two key observations: a) A page spans many flash cells and as soon as only one of the cells in a page reaches a state where it needs to be erased the entire page becomes unusable for rewriting. b) Flash drives incorporate ECC for each page and that this ECC is designed conservatively for the high bit error rates expect at the end of a drive's life, i.e. for much of the drive's life (when bit error rates are still low) the ECC could handle significantly larger error rates than what it actually experiences.

Our idea is to allow further rewrites of pages where only some limited number of cells have reached the V-MAX by marking those cells as invalid and let ECC recover the value in those cells. Considering random data bits, it is easy to see the variance in stored voltage levels across different cells in a page increases with the number of generations designed in the WOM-v code. While each WOM-v code certainly guarantee at least GEN_MAX number of possible write cycles in each cell, one could see that for a WOM-v code with larger GEN_MAX, there is a good chance to have majority of the cells still being capable of accommodating more reprograms after write cycle GEN_MAX. In order to enable writing beyond the write cycle GEN_MAX, we suggest to identify the cells not writable anymore by increasing the voltage level in them to V-MAX which would be interpreted as *invalid* for write cycles beyond GEN_MAX. As long as the percentage of such cells remains low in each page, an slightly stronger error correcting code (ECC) could handle these cells as noise, and enable reading stored data from the page.

Current literature on QLC flash drive reliability [25] provides limited insight to the existing ECC within a QLC drive. Given the increasing trend in error rates with denser media, we hypothesize QLC drive to have provisions for strong ECC mechanisms to correct uncorrectable bit errors. We suggest to use an enhanced ECC mechanism, rather than the already existing ECC embedded in the flash drives to correct raw bit errors in the SSDs. Since invalid cells in our proposed scheme would be identified as cells reaching V-MAX for write cycles beyond GEN_MAX, correcting this type of noise is easier than other types such as *rotten bits* caused by retention or program interference errors. This is because unlike traditional errors or corruptions that require reading parity or checksums for identifying the location of corrupted or errored cell, our approach simply checks for the value in a cell reaching the V-MAX voltage level.

Note that the maximum voltage level is marked as invalidated or EINVAL *after* the number of write cycles become more than GEN_MAX. This gives us a guaranteed GEN_MAX number of write cycles even without using ECC. Since all pages in a block undergo same number write cycles before erase, keeping count of write cycles has minimal storage overhead. Using pre-existing flash ECC to correctly reconstruct cells in V-MAX after write cycles reach GEN_MAX helps reprogram under-utilized cells which are still at lower Generations. By correcting cells that cannot be reprogrammed further, we increase the number of write cycles. For example, in WOM-v(1,4), increasing the number of writes helps increase logical space from 15X to 20X with a 4X physical space overhead. This raises the total logical writable bytes from 375% to 500% the original physical storage capacity of the drive.

Each of the three optimizations - code-word sharing, same generation transition and reusing underutilized cells, may be done for TLC drives as well, but the resultant gains will not be significant. However, as the disks become denser (eg. PLC) which is an upcoming trend for SSDs, our optimizations will reap great benefits while using WOM codes.

4 Evaluation

Our WOM coding scheme is based on simple table lookups and does not involve large firmware changes or complicated computation. In this section, we provide an analysis of the tradeoffs involved in implementing our scheme for QLC drives.

P/E Cycle versus Physical Space Tradeoff

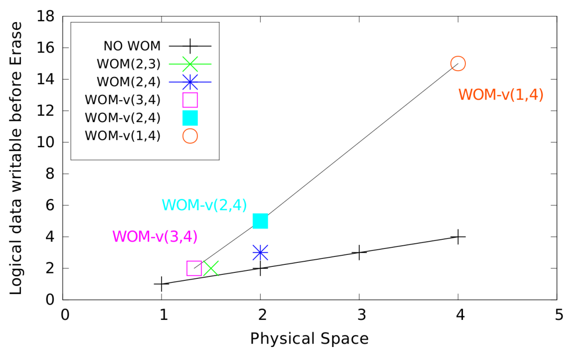

Figure 3: *Each point represents physical space required for 1 unit of user data, and corresponding logical writes possible before erase.*

Each WOM coding scheme offers a trade-off between number of write cycles possible before erase and the physical space overhead. For instance, in WOM(2,3) (Table 1), if a user has 1 unit of physical storage, we may write at least 2X logical data before erase is required. However, for each 2-bit data sequence, a 3-bit *code-word* is written, increasing storage overhead by 50%. While this might seem like a large space overhead, let's compare it with a solution that does not use codes, and therefore needs to perform an erase after every write. The only way to double the number of writes that is feasible for such a device is to double the physical capacity and spread all writes and erases over twice as many cells. This would be an additional 100% physical storage overhead.

Figure 3 shows the increased logical data before an erase versus the physical space overhead. With no coding NO WOM(+), the total amount of logical space written equals the total amount of physical space available. For more logical writes with a constant number of P/E cycles, the physical space required needs to be increased proportionately. For a 2X increase in logical writes using WOM(2,3)(X), we only need 1.5X physical space. Similarly, using WOM(2,4)($\star$), one can write 3X more logical data, with a space overhead of 2X. For WOM-v codes, although we need 1.33X, 2X and 4X physical space, we are able to write 2X, 5X and 15X amount of logical data for WOM-v(3,4), WOM-v(2,4) and WOM-v(1,4) respectively, before any cell reaches V-MAX. This is a huge gain over existing WOM(2,4) coding scheme that gives us a 50% additional logical space per physical unit. Using WOM-v(1,4) scheme, we get 375% additional logical space per physical unit. The lines joining WOM-v points show that underlying firmware may choose to take a middle-ground and use a combination of multiple codes based on available physical storage.

For example, if a firmware encodes half of the user data using WOM-v(2,4) and the other half using WOM-v(1,4), the overall physical space overhead per 1 unit of user facing physical storage equals $0.5 \times 2 + 0.5 \times 4 = 3$ units. The corresponding logical data that can be written before we need an erase operation would be $0.5 \times 5 + 0.5 \times 15 = 10$ units increasing overall logical space usage of physical media before erase by 333%.

Reprogramming beyond GEN_MAX

Once a cell in a page reaches V-MAX it can no longer be reprogrammed. The number of cells that have reached V-MAX can be easily computed in practice by reading cell values. In order to determine the probability of a cell reaching V-MAX, we use a Markov model considering each *data bit sequence* that is written to be an Independent and Identically Distributed (IID) random variable. At each voltage level, a transition to the same *code-word* or one of the other valid *code-words* takes place with equal probability. Therefore, in the worst case a cell will become unwritable only if each write cycle (WC) writes different data GEN_MAX times. Recall that after WC equals GEN_MAX, we treat V-MAX as EINVAL and treat all cells having EINVAL value unwritable. A coding scheme with larger GEN_MAX has less unwritable cells when WC equals GEN_MAX.

Figure 4 shows the amount of logical data that can be written for a given fraction of cells in EINVAL state after each WC for two coding schemes. We observe that for the same percentage of cells (0.02) in EINVAL, we are able to write more logical data beyond each code's respective GEN_MAX with WOM-v(1,4) as compared to WOM-v(2,4). We propose that if the fraction of cells reaching EINVAL beyond GEN_MAX is low (Eg. 0.02) and correctable using ECC, we can enhance the number of logical bytes written from 5X to 7X and from 15X to 20X in WOM-v(1,4) and WOM-v(2,4) respectively.
To analyze the total number of cells in a flash page that reaches EINVAL after GEN_MAX write cycles, we create a binomial distribution for each WC as shown in Figure 5. We consider the flash page size to be 4096. The X axis denotes the number of cells (k) in a flash page (between 0 and 4096). The Y axis denotes the probability of k cells having EINVAL value. For each WC, we provide the minimum correctable error rate (superscript on each curve) that needs to be maintained in flash ECC to correct cells in EINVAL. As the number of WC increase, the total number of cells in EINVAL in a page increase, which requires a stronger ECC.

We leave determining the exact value of ECC to the system designer based on the underlying media being used, as the uncorrectable bit error rate varies widely across flash media [25]. Rather, we provide an estimate of the ECC required to recover data in cells that have reached EINVAL after specific number of write cycles. For example, an ECC with error correction capability of 1/35 would help us do WC 7 and WC 20 increasing the logical bytes writable before erase from earlier 250% to 350% for WOM-v(2,4) and from earlier 375% to 500% for WOM-v(1,4) respectively.

5 Related Work

Flash memory reliability has been well studied in the past [6, 12, 14, 16, 17, 20, 21, 23, 28, 32]. These studies have become more relevant with denser media such as QLC drives that are less reliable and have decreased lifetime [25]. WOM Coding schemes [8, 22] have been evaluated for MLC and TLC flash [27, 28, 32]. The implementation and limitations of such approaches considering hardware constraints have been evaluated [18, 19, 33]. Further, biasing code-words towards one type of data format as opposed to another to keep voltage levels to a minimum have been studied [30, 35]. Recent work also suggests how encoding data such that eventual code is kept at a lower or middle voltage level helps reduce program interference errors [26, 30, 35]. Such approaches can be made complimentary to our approach as the voltage based model also tries to increase voltage by a minimum value during each write cycle and therefore keeps most cells at a lower/middle voltage level.

6 Future Work and Conclusion

We present a novel family of WOM codes to increase QLC flash lifetime. Our scheme is based on simple table lookups and does not involve complicated changes in underlying SSD firmware. We plan to test these models in field on real QLC flash chips and run real world workloads to get more accurate

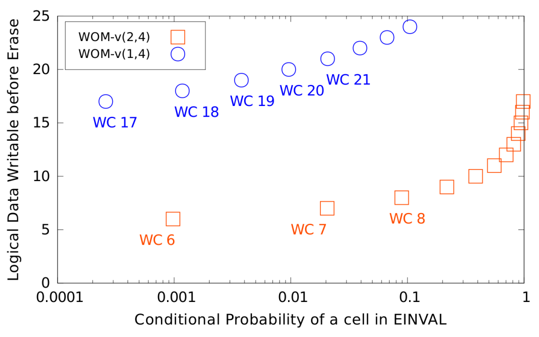

Figure 4: *Number of cells that reach EINVAL versus Logical Data Writable before erase. Each point shows the conditional probability of a cell being in V-MAX and the Write cycle (WC). Each additional WC increases Logical data writable before erase by 1. The probability of the cell in EINVAL is 5 and 16 for WOM-v(2,4) and WOM-v(1,4), not shown in figure due to log scale on the X-Axis. The probability is conditional based on a cell having a lower voltage than EINVAL.*

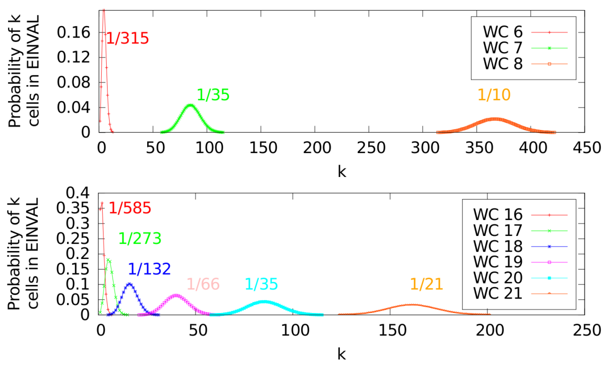

Figure 5: *Binomial distribution of the number of cells reaching EINVAL in a 4096 cell page for each write cycle for WOM-v(2,4) and WOM-v(1,4). The X axis denotes the number of cells (k) in a flash page. The Y axis denotes the probability of k cells having EINVAL value. 1/p denotes 1 additional ECC correction bit that needs to be maintained for every p data bits to continue reprogramming underutilized cells WC times.*

estimates of improvement in flash lifetime. We plan to evaluate the impact on performance caused by reading previously written data, reading and writing additional pages and saving time consuming erase operation during garbage collection in the future. Our solution is generic and can work for any N-Level Cell SSD.

7 Discussion

A new family of WOM codes for high density flash drives leads to some open questions for future research:

1. **Using Over-provisioned Space v/s ECC** Once flash cells start reaching EINVAL state, one way to not rely on ECC would be to write data of any unprogrammable cell to the over-provisioned space in the SSD. This brings additional management overhead but does not depend on internal ECC. This also improves the read performance as we no longer use ECC to correct the cells in EINVAL.
2. **Existing QLC RBER** During our preliminary discussions with some industry partners, we observed that there is a large variation in accepted values of RBER (Raw Bit-Error Rate) between different SSD vendors and across different SSD types. More discussion should be done to determine what is the industry accepted range of RBER rates for QLC flash and what error rates should academics assume for future reliability based research on dense flash storage media.
3. **Voltage Transition Limitations** Some voltage levels may be reserved. Our WOM code model can discard such reserved voltage levels while dividing different voltage levels into generations. Hence, our model would continue to work irrespective of forbidden voltage levels in hardware.
4. **Differential error rates at different voltage levels** All voltage thresholds are not equally error prone, and some states may have a higher error rate than others [26, 35]. One approach to implement WOM-v scheme would be give a larger voltage range between error prone voltage states. In voltage based model this can be achieved by merging multiple voltage states at a higher voltage level together and assigning a single code word to that voltage range.
5. **Re-programming Feasibility** Previous reliability studies attribute erase operation as a primary cause for flash wear. WOM codes involve reprogramming the same cell multiple times. It would be interesting to discuss the impact of re-programming on flash wear and if we should take the wear into account while quantifying increased QLC flash lifetime, as discussed for MLC flash in [31, 34].
6. **Hardware Tool Kits for QLC experiments** There exist limited support for QLC flash in open sourced tool kits and simulators [1, 3] to do benchmarks and experiments. It would be interesting to investigate some industry standard tool kits available for performing experiments, and get hardware level estimate of gains with the new class of WOM codes.
7. **Impact on simplifying SSD circuit design** Our WOM coding scheme restricts the number of transitions possible from one code word to another by 1/number of generations. For example, with NO WOM, a voltage may transition to one of the 16 transition levels. However, with WOM-v(2,4), we may only transition to the next 4 voltage levels. A future line of research could be towards simplifying SSD circuitry.

References

[1] Open-source solid-state drive project for research and education. http://openssd.io. Accessed: 2020-03-24.

[2] Western digital and toshiba talk up penta-level cell flash. https://blocksandfiles.com/2019/08/07/penta-level-cell-flash/. Accessed: 2020-03-24.

[3] Xylinx boards and kits. https://www.xilinx.com/products/boards-and-kits.html. Accessed: 2020-03-24.

[4] Nicolas Bitouzé, Alexandre Graell i Amat, and Eirik Rosnes. Using short synchronous wom codes to make wom codes decodable. *IEEE transactions on communications*, 62(7):2156–2169, 2014.

[5] Sarit Buzaglo and Tuvi Etzion. Tilings with n-dimensional chairs and their applications to asymmetric codes. *IEEE transactions on information theory*, 59(3):1573–1582, 2012.

[6] Yu Cai, Saugata Ghose, Erich F Haratsch, Yixin Luo, and Onur Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE*, 105(9):1666–1704, 2017.

[7] Yuval Cassuto and Eitan Yaakobi. Short q-ary fixed-rate wom codes for guaranteed rewrites and with hot/cold write differentiation. *IEEE transactions on information theory*, 60(7):3942–3958, 2014.

[8] Amos Fiat and Adi Shamir. Generalized'write-once'memories. *IEEE Transactions on Information Theory*, 30(3):470–480, 1984.

[9] Hilary Finucane, Zhenming Liu, and Michael Mitzenmacher. Designing floating codes for expected performance. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 1389–1396. IEEE, 2008.

[10] Fang-Wei Fu and AJ Han Vinck. On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph. *IEEE Transactions on Information Theory*, 45(1):308–313, 1999.

[11] Ryan Gabrys and Lara Dolecek. Constructions of non-binary wom codes for multilevel flash memories. *IEEE Transactions on Information Theory*, 61(4):1905–1919, 2015.

[12] Laura M Grupp, Adrian M Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H Siegel, and Jack K Wolf. Characterizing flash memory: anomalies, observations, and applications. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 24–33. IEEE, 2009.

[13] Qin Huang, Shu Lin, and Khaled AS Abdel-Ghaffar. Error-correcting codes for flash coding. *IEEE transactions on information theory*, 57(9):6097–6108, 2011.

[14] Shehbaz Jaffer, Stathis Maneas, Andy Hwang, and Bianca Schroeder. Evaluating file system reliability on solid state drives. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 783–798, Renton, WA, July 2019. USENIX Association.

[15] BM Kurkoski. Notes on a lattice-based wom construction that guarantees two writes. In *Proc. 34th Symp. Information Theory and Applications*, pages 520–524, 2011.

[16] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. Improving 3d nand flash memory lifetime by tolerating early retention loss and process variation. In *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '18, pages 106–106, New York, NY, USA, 2018. ACM.

[17] Stathis Maneas, Kaveh Mahdaviani, Tim Emami, and Bianca Schroeder. A study of SSD reliability in large scale enterprise storage deployments. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 137–149, Santa Clara, CA, February 2020. USENIX Association.

[18] Fabio Margaglia and André Brinkmann. Improving mlc flash performance and endurance with extended p/e cycles. In *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–12. IEEE, 2015.

[19] Fabio Margaglia, Gala Yadgar, Eitan Yaakobi, Yue Li, Assaf Schuster, and Andre Brinkmann. The devil is in the details: Implementing flash page reuse with WOM codes. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 95–109, Santa Clara, CA, February 2016. USENIX Association.

[20] Justin Meza, Qiang Wu, Sanjev Kumar, and Onur Mutlu. A large-scale study of flash memory failures in the field. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):177–190, 2015.

[21] Iyswarya Narayanan, Di Wang, Myeongjae Jeon, Bikash Sharma, Laura Caulfield, Anand Sivasubramaniam, Ben Cutler, Jie Liu, Badriddine Khessib, and Kushagra Vaid. Ssd failures in datacenters: What? when? and why? In *Proceedings of the 9th ACM International on Systems*

and Storage Conference, SYSTOR '16, New York, NY, USA, 2016. Association for Computing Machinery.

[22] Ronald L Rivest and Adi Shamir. How to reuse a "write-once memory. *Information and control*, 55(1-3):1–19, 1982.

[23] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash reliability in production: The expected and the unexpected. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 67–80, Santa Clara, CA, February 2016. USENIX Association.

[24] Amir Shpilka. Capacity-achieving multiwrite wom codes. *IEEE Transactions on Information Theory*, 60(3):1481–1487, 2013.

[25] Amy Tai, Andrew Kryczka, Shobhit O Kanaujia, Kyle Jamieson, Michael J Freedman, and Asaf Cidon. Who's afraid of uncorrectable bit errors? online recovery of flash errors with distributed redundancy. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 977–992, 2019.

[26] Debao Wei, Libao Deng, Peng Zhang, Liyan Qiao, and Xiyuan Peng. Nrc: A nibble remapping coding strategy for nand flash reliability extension. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1942–1946, 2016.

[27] Eitan Yaakobi, Laura Grupp, Paul H Siegel, Steven Swanson, and Jack K Wolf. Characterization and error-correcting codes for tlc flash memories. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 486–491. IEEE.

[28] Eitan Yaakobi, Jing Ma, Laura Grupp, Paul H Siegel, Steven Swanson, and Jack K Wolf. Error characterization and coding schemes for flash memories. In *2010 IEEE Globecom Workshops*, pages 1856–1860. IEEE, 2010.

[29] Eitan Yaakobi and Amir Shpilka. High sum-rate three-write and nonbinary wom codes. *IEEE Transactions on Information Theory*, 60(11):7006–7015, 2014.

[30] Eitan Yaakobi, Gala Yadgar, Nachum Bundak, and Lior Gilon. A case for biased programming in flash. In *10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18)*, 2018.

[31] Gala Yadgar and Roman Shor. Experience from two years of visualizing flash with ssdplayer. *ACM Trans. Storage*, 13(4), November 2017.

[32] Gala Yadgar, Eitan Yaakobi, Fabio Margaglia, Yue Li, Alexander Yucovich, Nachum Bundak, Lior Gilon, Nir Yakovi, Assaf Schuster, and André Brinkmann. An analysis of flash page reuse with wom codes. *ACM Transactions on Storage (TOS)*, 14(1):1–39, 2018.

[33] Gala Yadgar, Eitan Yaakobi, and Assaf Schuster. Write once, get 50% free: Saving SSD erase costs using WOM codes. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 257–271, 2015.

[34] Gala Yadgar, Alexander Yucovich, Hila Arobas, Eitan Yaakobi, Yue Li, Fabio Margaglia, André Brinkmann, and Assaf Schuster. Limitations on mlc flash page reuse and its effects on durability. Technical report, Computer Science Department, Technion, 2016.

[35] Yutong Zhao, Wei Tong, Jingning Liu, Dan Feng, and Hongwei Qin. Cesr: A cell state remapping strategy to reduce raw bit error rate of mlc nand flash. In *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 161–171. IEEE, 2019.