Multi-page read for NAND flash

Tianqiong Luo and Borja Peleato

***Abstract*—NAND flash memories achieve very high densities through a series connection of all the cells in a bitline. In current memories, each wordline is read independently by biasing all the other cells to act as pass transistors and sensing all the bitlines in parallel. This paper proposes a new method which reads multiple wordlines simultaneously and returns a combination of their stored information. This multi-page read method is shown to be useful for equalizing the inter-cell interference, reduce the damage caused by erase operations, and speed up the decoding of a certain class of codes.**

***Index Terms*—NAND flash memory, ICI equalization, WOM codes, memory architecture**

I. Introduction

A flash memory is fundamentally an array of floating gate transistors, known as flash cells, whose threshold voltages can be programmed to represent different information symbols. Single-Level Cell (SLC) memories can only store one bit in each cell, but most commercial products now use Multi-Level Cell (MLC) memories that can store two bits in each cell by taking 4 possible voltages. Flash manufacturers have tried to increase the capacity of the memories by shrinking the cells and storing more bits in each of them but this has introduced some challenges, mainly related to reliability and endurance.

As the cells shrink, the noise observed in the programmed voltages increases, specially the inter-cell interference (ICI). ICI is a phenomenon by which programming a cell increases the voltage of its neighbors. It has been shown that the ICI noise created by a cell is proportional to the voltage to which it is programmed [1].

Another challenge that the flash memory industry is facing is the limited lifetime of flash cells. Basic operations like programming and erasing[1] the cells require tunneling charges through a dielectric barrier. This results in stresses that degrade the properties of the barrier making the cells less efficient in the retention of data, more vulnerable to noise, and consequently more prone to errors. Once again, the degradation is proportional to the voltage variations [2].

This paper proposes a new read method, which we call multi-page read, that can help alleviate some of the challenges that the flash memory industry is facing. A multi-page read operation selects multiple pages in a block, biases them with different read thresholds, and returns a combination of their stored information. The definitions for page, block, and read threshold will be given in Section II, together with some necessary background on NAND flash memories. Section III describes the multi-page read and explains how it should be implemented. Then, Section IV provides examples where the multi-page read can help improve the reliability, speed, and endurance of Flash memories. Finally, Section V concludes the paper.

T. Luo and B. Peleato are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907. (e-mail: luo133@purdue.edu; bpeleato@purdue.edu)

[1]Flash cells need to be erased before they can be overwritten.

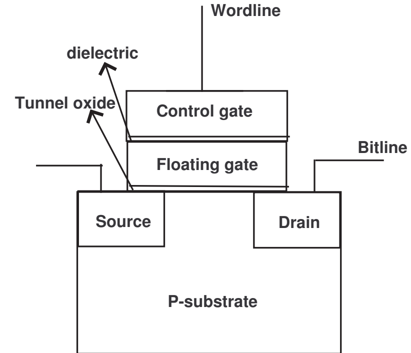


Fig. 1: Floating gate transistor.

II. Background

A flash cell, illustrated in Fig. 1, is a floating gate transistor whose threshold voltage can be adjusted by Fowler-Nordheim (FN) tunneling [3] of charge into or out of the floating gate. If the control gate voltage is greater than this threshold, the cell opens a channel between the drain and the source and we say that the cell *conducts*. Otherwise, the cell acts as an open circuit and the cell does not conduct. NAND flash memories organize cells in array structures known as blocks, like the one shown in Fig. 2. We refer to each row of cells in a block as a wordline and to each column as a bitline. A page is a logical structure that includes one bit from each cell in a wordline. SLC memories have one page per wordline, MLC memories have two, and TLC have three. Blocks are the elementary unit for erase operations, but reads and writes can be done at a page granularity. This wordline-bitline structure allows programming or reading all the cells in a page in parallel, as described below.

1) Program operation: The programming is done by sending high voltage pulses into one wordline and biasing all other wordlines so that their cells conduct. Cells in the selected wordline with grounded bitlines experience a high electric field across the floating gate and the substrate, triggering the FN tunneling. After each pulse, a verify read is performed and cells which have reached the desired level of charge are inhibited from further programming. This can be done by biasing their bitlines to a high voltage. This programming method is called ISPP algorithm [4]. For MLC cells, the programming includes two stages: the LSB programming leaves the cell either erased or at half its maximum charge, and the MSB programming does the fine adjustment of the final

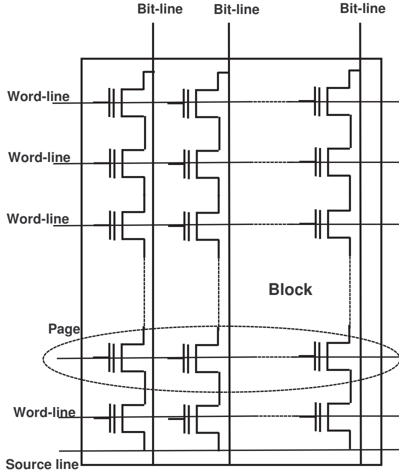


Fig. 2: Bitline-Wordline structure of flash memory.

voltage. The amount of electrons injected into the floating gate is determined by both the LSB and MSB bit values [5].

2) Erase operation: The erasing of a block follows an similar process to the programming, but using negative pulses to remove charges from the floating gate instead. Additionally, stronger and fewer pulses are used since there is little harm in over-erasing the cells. Individual pages cannot be erased independently because a dielectric breakdown may occur due to the interference between wordlines [6].

3) Read operation: The voltage threshold of the cells cannot be read directly, it can only be compared with an adjustable reference (read voltage). Pages are read by biasing one wordline with this read voltage l while all others are set to a high voltage V_{pass} ($l \ll V_{\text{pass}}$) so that their cells conduct. Cells in the selected wordline whose threshold voltage is below the read voltage l also conduct, causing the discharge of a capacitor through the bitline, whereas cells with higher threshold voltage act as an open circuit, not letting the current through. By sensing which capacitors got discharged, many bitlines can be read in parallel.

III. Multi-page read method

The read operation described above provides one bit of information about each cell in the selected wordline. If a bitline conducts, it means that the cell's threshold voltage is below the read voltage. The corresponding bit is then read as "0". If a bitline does not conduct, it means that the cell's threshold is above the read voltage and the corresponding bit is then read as "1". However, it is important to understand that the bit values depend on the read threshold: the same page can yield different bit values for different read thresholds[2].

From the perspective of sensing circuits, there exist multiple read architectures, all of which use capacitances to integrate the bitline current. Most modern NAND Flash memories use the All Bitline (ABL) architecture [7] shown in Fig. 4, which includes a dedicated capacitor C_{SO} and keeps the bitline

[2]Some manufacturers use the reverse bit labels, "1" to denote conducting and "0" not conducting. This convention makes no difference towards our results, but OR operations should be replaced with AND.

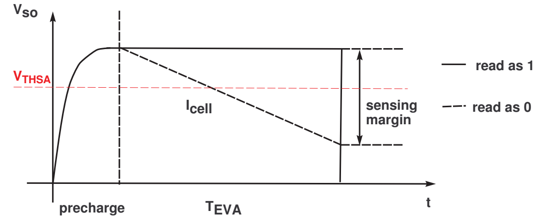


Fig. 3: ABL read operation timing diagram.

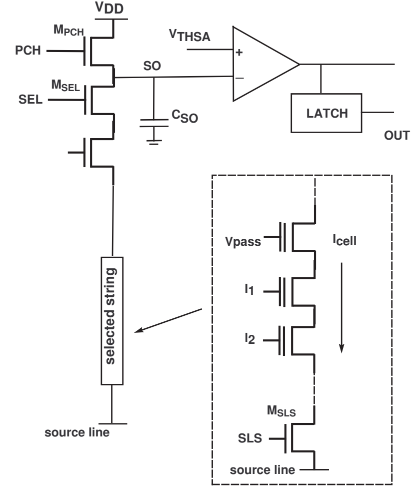


Fig. 4: ABL sense circuits for NAND flash.

voltage constant during the evaluation phase. Fig. 3 shows the three phases in a read operation. First, C_{SO} is pre-charged to a high voltage V_{DD}. Then M_{PCH} is shut off and conducting bitlines experience a constant current I_{cell} that discharges C_{SO} (evaluation phase). After T_{EVA} seconds, the capacitor voltage is compared with a reference V_{THSA} and the read result is output through a latch [8]. The total read time is dominated by the evaluation time T_{EVA}, which can be represented as:

$$T_{\text{EVA}} = \frac{(V_{\text{DD}} - V_{\text{THSA}})C_{\text{SO}}}{I_{\text{cell}}}. \tag{1}$$

The current through a MOS transistor operating in ohmic region I_{cell} with small drain to source voltage V_{DS} can be approximated by:

$$I_{\text{cell}} = k\left[(V_{\text{GS}} - V_{\text{TH}})V_{\text{DS}}\right], \tag{2}$$

where V_{GS} and V_{TH} respectively represent the gate-to-source and threshold voltages and k is a scaling parameter [8]. Hence, the equivalent resistance R_{oh} for the transistor working in the Ohmic region is:

$$R_{\text{oh}} = \frac{V_{\text{DS}}}{I_{\text{cell}}} = \frac{1}{k(V_{\text{GS}} - V_{\text{TH}})}. \tag{3}$$

The multi-page read proposed in this paper uses the same components and read methodology as the ABL architecture, but instead of biasing a single wordline with a read threshold and all others with V_{pass}, multiple wordlines are biased with different read thresholds $\{l_1, l_2, \ldots\}$ while the rest are kept at V_{pass} as shown in Fig. 4. A bitline will conduct only when all the selected cells have lower voltage than the corresponding

read thresholds. Since we are using value "1" to denote "not conducting", this is equivalent to a bit-wise OR operation of all the selected wordlines.

The main problem of the ABL architecture is the static current consumption during the pre-charge phase, specially by cells with threshold voltage much smaller than the read voltage as indicated by Eq. (2). With multi-page read, however, fewer bitlines will conduct and those that do will only draw strong currents if **all** the read voltages are much larger than the corresponding thresholds. Hence, multi-page read helps alleviate the power consumption problem.

Bitlines that do conduct will experience a read current very similar to that in regular reads. Each bitline has hundreds of cells connected in series whose equivalent resistance is determined by the gap between the read and threshold voltages according to Eq. (3). This gap is smaller for cells being read than for those biased at V_{pass}, but both are usually in the same order of magnitude. The equivalent resistance of the whole string is then dominated by the hundreds of cells acting as pass transistors, not by the few being read. If the read and threshold voltages are very close for a cell, it would reach the saturation mode, thereby limiting the current independently of V_{DS}. Since I_{cell} in Eq. (1) does not suffer a significant decrease in either case, we can conclude that the evaluation time T_{EVA} in a multi-page read is similar to that in a regular read, offering comparable read speeds.

Additionally, the multi-page read method can be applied to improve several applications of flash memories as we will discuss in Section IV.

IV. Applications

A. ICI Equalization

The ISPP programming algorithm can compensate for the inter-cell interference caused by previously programmed wordlines, but not for the interference of subsequent write operations. Since wordlines are programmed in sequential order, most of the ICI suffered by a specific wordline is caused by the direct-above-neighbor. Extensive measurements have shown that the change in threshold voltage suffered by the victim cell is proportional to the threshold voltage of the aggressor cell, with a proportionally factor γ that depends on the parasitic capacitance between the aggressor cell and the victim cell.

The neighbor-cell assisted error correction (NAC) algorithm was proposed in [5] to equalize ICI. The NAC method first performs one read of the aggressor wordline and classifies the cells in the victim wordline as suffering weak or strong ICI depending on the value programmed in their direct above neighbor. Then, it reads the victim wordline with different thresholds, selectively chosing which result to keep for each cell. It effectively reads cells suffering strong ICI with a different threshold as those suffering weak ICI, thereby reducing the probability of error. However, this algorithm requires reading the aggressor wordline and thus reduces the read speed. We propose to use the multi-page read method to read the victim and aggressor wordlines simultaneously.

If we set a read threshold l_{victim} on the desired wordline and an intermediate threshold $l_{\text{aggressor}}$ on its neighbor, while the

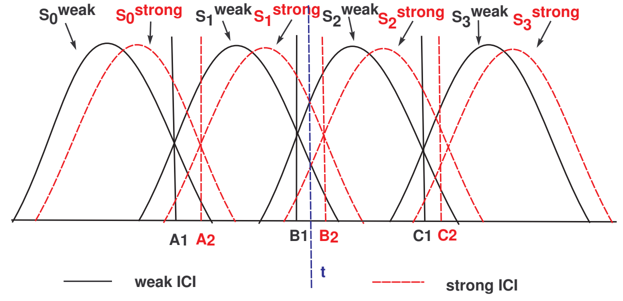
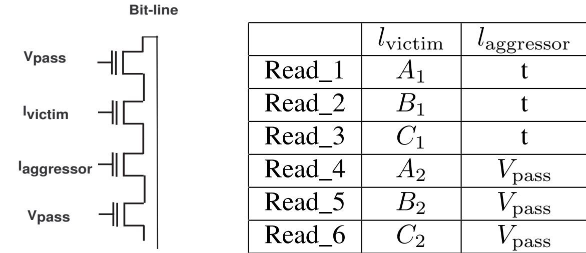


Fig. 5: Illustration of multi-page read method for MLC ICI equalization. The curves represent histograms of threshold voltages across a page and vertical lines represent read thresholds.



	l_{victim}	$l_{\text{aggressor}}$
Read_1	A_1	t
Read_2	B_1	t
Read_3	C_1	t
Read_4	A_2	V_{pass}
Read_5	B_2	V_{pass}
Read_6	C_2	V_{pass}

Fig. 6 & TABLE I: Bitline illustration & Multi-page reads for MLC ICI equalization.

rest are set to V_{pass}, only bitlines which fulfil both conditions would conduct. This way we can use a single multi-page read to detect the cells which have voltage below l_{victim} **and** are suffering weak ICI. Combining these multi-page reads allows us to obtain similar results to the NAC algorithm.

For example: In MLC memories, each cell can be programmed to four different levels, denoted S_0, S_1, S_2, and S_3. According to the suffered ICI, we further classify them into 8 states: S_0^{weak}, S_0^{strong}, S_1^{weak}, $S_1^{\text{strong}}\ldots$, as Fig. 5 shows. The read thresholds for weak ICI cells are A_1, B_1, C_1 and for strong ICI cells A_2, B_2, C_2. The six proposed reads are listed in Table I.

To classify the cells into S_0, S_1, S_2 and S_3, we combine the results of the six threshold comparisons. State S_i $(i = 0,1,2,3)$ can be represented as $(S_i^{\text{weak}}$ OR $S_i^{\text{strong}})$, where S_i^{weak} can be found from the first three reads and S_i^{strong} can be found from the last three reads after eliminating weak ICI cells. For example, we classify a cell as S_1 iff {Read_1=0 and Read_2=1} OR {Read_4=0 and Read_5=1 and Read_3=0}.

This strategy brings slightly more error between S_2 and S_3 than NAC, which performs an additional read on the aggressor wordline. In order to reduce this error C_2 is slightly shifted, as shown in Fig. 5. However, the error is minor and this strategy will save one read. Moreover, when ICI from the nearest neighbor is large, this method provides lower BER than any other with reads on a single page ever could.

Fig. 7 compares the channel capacity of a regular scheme using six reads to produce soft information [9] with NAC and the multi-page read method for MLC flash memories. It is assumed that S_0, S_1, S_2, S_3 are Gaussian distributed [1] with

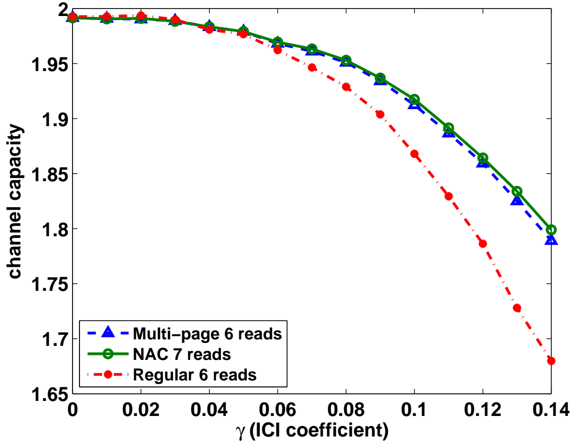


Fig. 7: Channel capacity for an MLC cell after 6 traditional reads, 6 multi-page reads, and NAC with 7 reads.

the same variance $\sigma = 0.15$ and means $0, 1, 2, 3$, respectively. The read thresholds for all three methods were numerically optimized to maximize the resulting capacity. The results show that as γ increases, the NAC and multi-page read method provide significantly better performance. In fact, when ICI dominates over the Gaussian noise, the proposed method would always provide higher capacity than reading the desired page alone, regardless of how many reads the latter employs. This is due to the fact that the multi-page read method provides some amount of equalization for the channel. The figure also shows that the performance of multi-page read is very close to that for the NAC method despite it requires one less read operation.

B. Partial Erase

The erase operation consists of sending a series of voltage pulses into the gates of all the cells in a block, until all the floating gates have been discharged [10]. All cells in the block suffer the same pulses, despite some can be "fast erased" and others need more negative pulses [11]. These pulses damage the dielectric barrier in the cells, increasing BER and shortening their lifetime. This subsection shows how to apply the multi-page read method to reduce the number of erase pulses sent, so that this damage can be reduced and the erase operation can be accelerated.

Flash memories generally use a log-structured file system, with a background garbage collection process that keeps a pool of erased blocks ready to be written [12]. As new information arrives, the controller writes it in these blocks, filling one before moving on to the next.

Unlike the traditional erase algorithm, which continues issuing pulses until the whole block is totally erased, we propose a partial-erase option where fewer pulses are sent and a small number of cells remain incompletely erased. By reading all the wordlines simultaneously with a very low read threshold, the controller can detect which bitlines have cells that have not been completely erased, and store their indices as part of the block's metadata. The controller may then chose to skip those bitlines during the writing process, use a constraint code to mask the errors [13], or ignore this information and rely on the ECC block to correct any errors that might arise.

Similar ideas can also be applied to worn-out flash memories. Although it is common to assume a uniform wear among all the cells, not all of them present the same level of tolerance towards program-erase (P/E) cycles. This makes the reuse of worn-out blocks meaningful. Lab collected data shows that erase errors, which typically trigger the permanent retirement of a block, are caused by a few broken cells that remain unerased, but most of the other cells are healthy. By setting all wordlines to a small read threshold, we can detect which bitlines have broken cells and skip them in subsequent writes.

Unfortunately, the multi-page read step may slow down the erase operation. The gap between the read and threshold voltages could be relatively small for many cells along the bitline, thereby limiting the current and extending the evaluation time. However, the read step still takes much less time than sending the erase pulses. According to [14], the erase operation for one block takes about $500\mu s$ while the read operation takes only several nanoseconds. So the latency brought by the multi-page read step is negligible in the partial erase operation.

C. WOM Codes

WOM (Write-Once-Memory) codes were designed for memories where bits can change in one direction (*e.g.*, 0 to 1) but not the other [15], [16], and they have recently been proposed to allow multiple overwrites of a flash page without erasing [17]. A simple example of a WOM code is shown in Fig. 8. This example is using three SLC (binary value) cells to write two bits twice. Initially, all three cells are erased (state 000). The first two bits are written by transitioning to one of the states in the first generation, according to the labels shown in the plot. The second pair of bits, if different from the first, is written by transitioning to one of the states in the second generation. All transitions involve charging, not erasing, the cells so they are feasible.

However, WOM codes present some practical limitations that prevent their adoption by the flash memory industry: they significantly reduce the capacity and speed of the memory. The example shown in Fig. 8 provides two bits of information for every three cells read, so page length and read throughput are 33% lower than with a traditional scheme.

In order to address these challenges, we propose aligning the WOM codewords vertically, across wordlines, instead of storing the whole codeword on the same page. The multi-page read allows us to rapidly compute the OR operation of multiple wordlines, accelerating the decoding. Observe that, denoting the state of the cells by $b_1b_2b_3$, the two bits in the first generation are given by "b_2 OR b_3" and "b_1 OR b_3", respectively. A single multi-page read of the last two wordlines would provide the first information bit and a single multi-page read of the first and third wordlines the second information bit, as shown in Fig. 9. Each multi-page read provides a sequence of information bits of the same length as a page, so the read throughput is the same as in the traditional scheme.

Unfortunately, a similar idea cannot be applied to the second generation of writes. Some codewords may still be in first

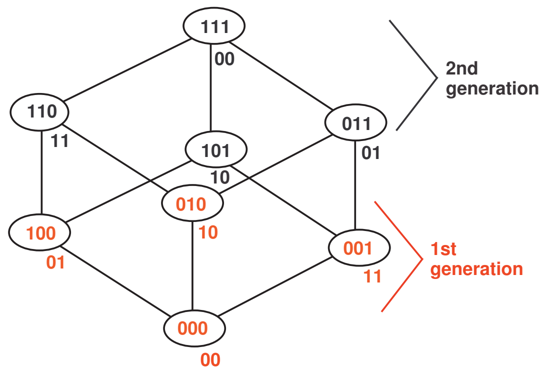


Fig. 8: The WOM code on the cube.

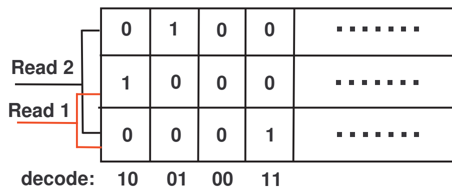


Fig. 9: Multi-page read to decode WOM code.

generation states so it is necessary to read all three pages individually to obtain two pages of information. This yields the same rate as the regular WOM scheme, which performs one read to obtain $\frac{2}{3}$ of a page of information.

On average, for the example shown in Fig. 8, the proposed scheme with multi-page reads would provide an information rate of 0.8 (4 pages of information after 5 reads), which is a significant improvement over the 0.66 rate shown by the usual WOM scheme. Similar approaches might be possible for more advanced WOM codes.

D. Other applications

The multi-page read provides a way of obtaining the bit-wise OR (or bit-wise AND, if the discharged state is denoted by 1) of the information stored in multiple wordlines using a single read operation[3]. There are multiple applications that could benefit from such feature: group testing, masking, constrained codes, hash lookups, etc. Instead of reading multiple pages and storing them in registers to perform these operations, the multi-page read allows us to obtain the result in a fraction of the time by performing a single read.

In this section we have shown some of the possibilities that this new read method can provide, but there are many others which we have not included in the paper. In further research, we plan to explore some of these applications in more detail.

V. Conclusion

A new read method for NAND flash memories called multi-page read method is proposed in this paper. This method reads multiple wordlines together and returns a bitwise OR (or AND, depending on notation) of their stored information. The examples provided in this paper show that this read method can improve the reliability of the stored information by equalizing ICI noise, reduce the damage caused by erase operations, and accelerate the decoding of certain constrained codes. Furthermore, the proposed read method provides a very fast way of finding the bitwise AND/OR of multiple codewords without having to read each operand separately, so it is very likely that there exist other applications that can be studied in future research.

[3]For MLC memories, finding the OR of an MSB page would require two multi-read threshold comparisons, just like in a traditional MSB read.

References

[1] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory," IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 57, no. 10, pp. 2718–2728, Oct. 2010.

[2] W. Wang, T. Xie, and D. Zhou, "Understanding the impact of threshold voltage on MLC flash memory performance and reliability," in Proc. 28th ACM Int. Conf. on Supercomputing (ICS). ACM, 2014, pp. 201–210.

[3] R. Katsumata, M. Kito, Y. Fukuzumi, M. Kido, H. Tanaka, Y. Komori, M. Ishiduki, J. Matsunami, T. Fujiwara, Y. Nagata et al., "Pipe-shaped BiCS flash memory with 16 stacked layers and multi-level-cell operation for ultra high density storage devices," in Proc. IEEE Symp. on VLSI Technol., 2009, pp. 136–137.

[4] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum et al., "A 3.3 v 32 mb NAND flash memory with incremental step pulse programming scheme," IEEE J. Solid-State Circuits, vol. 30, no. 11, pp. 1149–1156, Nov. 1995.

[5] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai, "Neighbor-cell assisted error correction for MLC NAND flash memories," in ACM Int. Conf. on Meas. and Modeling of Comput. Syst. ACM, 2014, pp. 491–504.

[6] J. E. Brewer and M. Gill, Nonvolatile Memory Technologies with Emphasis on Flash. Wiley, 2008.

[7] R. Cernea, L. Pham, F. Moogat, S. Chan, B. Le, Y. Li, S. Tsao, T.-Y. Tseng, K. Nguyen, J. Li et al., "A 34MB/s-program-throughput 16Gb MLC NAND with all-bitline architecture in 56nm," in IEEE Int. Solid-State Circuits Conf. (ISSCC), 2008, pp. 420–624.

[8] R. Micheloni, L. Crippa, and A. Marelli, Inside NAND Flash Memories. Springer, 2010.

[9] J. Wang, T. Courtade, H. Shankar, and R. Wesel, "Soft information for LDPC decoding in flash: Mutual information optimized quantization," in IEEE Global Communications Conf. (GLOBECOM). IEEE, 2011, pp. 5–9.

[10] G. Wu and X. He, "Reducing SSD read latency via NAND flash program and erase suspension," in Proc. of the 10th USENIX Conf. on File and Storage Technol., vol. 12, 2012, pp. 10–10.

[11] S. C. Hollmer, C.-Y. Hu, B. Q. Le, P.-l. Chen, J. Su, R. Gutala, and C. Bill, "Erase verify scheme for NAND flash," Dec. 28 1999, US Patent 6,009,014.

[12] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," ACM Computing Surveys (CSUR), vol. 37, no. 2, pp. 138–163, 2005.

[13] Y. Kim and B. V. Kumar, "Coding for memory with stuck-at defects," in IEEE Int. Conf. on Communications (ICC). IEEE, 2013, pp. 4347 – 4352.

[14] J. Cooke, "Flash memory 101: An introduction to NAND flash," Micron Technology Inc: Boise, 2006.

[15] R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory," Information and control, vol. 55, no. 1, pp. 1–19, 1982.

[16] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," IEEE Trans. Inf. Theory, vol. 58, no. 9, pp. 5985–5999, 2012.

[17] R. Gabrys and L. Dolecek, "Constructions of nonbinary WOM codes for multilevel flash memories," IEEE Trans. Inf. Theory, vol. 61, no. 4, pp. 1905–1919, Apr. 2015.